

Studie naar het ontwerp van 'DIY Social Robots'

Study on the Design of DIY Social Robots

Cesar Vandevelde

Promotoren: prof. dr. ing. J. Saldien, prof. dr. ir. F. wyffels  
Proefschrift ingediend tot het behalen van de graad van  
Doctor in de industriële wetenschappen: industrieel ontwerpen

Vakgroep Industriële Systemen en Productontwerp  
Voorzitter: prof. dr. E.-H. Aghezzaf

Vakgroep Elektronica en Informatiesystemen  
Voorzitter: prof. dr. ir. R. Van de Walle

Faculteit Ingenieurswetenschappen en Architectuur  
Academiejaar 2016 - 2017



UNIVERSITEIT  
GENT

ISBN 978-90-8578-987-1

NUR 964

Wettelijk depot: D/2017/10.500/22



# EXAMENCOMMISSIE

- Prof. Rik Van de Walle – *voorzitter*  
Vakgroep Elektronica en Informatiesystemen  
Universiteit Gent
- Prof. Tony Belpaeme – *secretaris*  
Vakgroep Elektronica en Informatiesystemen  
Universiteit Gent
- Prof. Jelle Saldien – *promotor*  
Vakgroep Industriële Systemen en Productontwerp  
Universiteit Gent
- Prof. Francis wyffels – *promotor*  
Vakgroep Elektronica en Informatiesystemen  
Universiteit Gent
- Dr. Ruben Verstraeten  
Vakgroep Architectuur en Stedenbouw  
Universiteit Gent
- Prof. Luc Geurts  
Technologiecluster Elektrotechniek  
KU Leuven
- Prof. Bram Vanderborght  
Robotics and Multibody Mechanics  
Vrije Universiteit Brussel



# DANKWOORD

Vier jaar geleden kreeg ik de unieke kans om een doctoraat te starten in het Industrial Design Center. Tijdens deze periode kon ik dingen doen waarvan de meeste startende ontwerpers alleen maar kunnen dromen: ik kon mij verdiepen in mijn onderwerp, ik kwam in contact met andere onderzoekers wereldwijd, en ik had de vrijheid om te experimenteren met de nieuwste technologieën. Het was niet alleen een kans om mij op technisch-wetenschappelijk vlak te ontplooiën, maar ook om mijzelf voor een stuk te ontdekken. Deze uitdaging ben ik niet alleen aangegaan, doorheen het volledige proces kon ik rekenen op de steun van anderen. Daarom wil ik graag iedereen bedanken met wie ik de afgelopen vier jaar samengewerkt heb.

In de eerste plaats wil ik graag mijn promotoren prof. Jelle Saldien en prof. Francis wyffels bedanken. Ze gaven me de vrijheid en het vertrouwen om op eigen tempo te exploreren. Ze reikten mij kansen en opportuniteiten aan. En vooral, ze geloofden in het potentieel van mijn werk zelfs wanneer ik het zelf niet meer zag. Ook mijn begeleiders Bram Vanderborcht en Maria-Cristina Ciocchi wil ik van harte bedanken. Altijd kon ik op hen rekenen voor een praktische vraag, voor feedback op een tekst, of gewoon voor een goed gesprek.

De voorbije jaren heb ik altijd het geluk gehad om omringd te zijn door uitstekende collega's. Mensen die gedreven zijn en fantastisch zijn in wat ze doen. Op de eerste plaats denk ik aan de collega's Industrieel Ontwerpen en Industrieel Productontwerpen: jullie stonden altijd klaar met een gezonde dosis inspiratie, humor, en relativiseringsvermogen. In het bijzonder wil ik Dries Bovijn en Stan Notebaert bedanken voor wat ze tot nu toe verwezenlijkt hebben voor de Opsoro spin-off. Ook oud-collega Maarten Vanhoucke ben ik dankbaar voor de leuke momenten in het ProtoLab en voor alle praktische hulp in drukke tijden. Daarnaast denk ik nog aan prototyping "goeroes" Bart Grimonprez en Carl Grimmelprez: bedankt om mij continu uit mijn comfort zone te sleuren door te tonen hoe breed prototyping kan zijn. Aan de mede-doctorandi binnen het IDC: Peter, Francesca, Bert, Davy, Bram: bedankt voor alles wat ik van jullie geleerd heb, bedankt voor jullie sublieme meesterschap van humor en zelfspot, en vooral bedankt voor alle fijne discussies tussen pot en pint. Tenslotte bedankt aan alle collega's van de opleiding electronica-ICT, met Benjamin Samyn en Xavier Vanhoutte in het bijzonder. Van hen heb ik enorm veel opgestoken wat betreft electronica-ontwerp. In tijden van nood kon ik altijd op hen rekenen om een schakeling te debuggen of een board te bestukken.

Doorheen het volledige onderzoekstraject van mijn doctoraat werden studentenopdrachten betrokken in het ontwerpproces. Aan alle studenten die aan één van deze opdrachten hebben meegewerkt: dank je voor jullie inzet, creativiteit, inspiratie, en enthousiasme! In het bijzonder wil ik graag IO-alumni Pieter-Jan Mollé en Marie Van den Broeck bedanken. Zij kozen om binnen hun masterproef verder te werken met Ono, en hebben op die manier een enorme bijdrage geleverd tot dit doctoraat.

Tenslotte wil ik mijn ouders, vrienden, en familie bedanken voor hun steun doorheen dit proces. Aan mijn ouders: jullie stonden altijd klaar met raad en daad, jullie hebben voor rust en vrijheid gezorgd zodat ik mij volledig op mijn doctoraat kon richten. Zonder jullie steun zou ik ongetwijfeld nooit zo ver geraakt zijn. Aan mijn vrienden: jullie waren altijd overtuigd dat ik deze uitdaging tot een goed einde zou brengen. Ook wanneer ik het meest aan mijzelf twijfelde bleven jullie in mij geloven. Het behalen van dit doctoraat is één van de moeilijkste uitdagingen die ik tot nu toe heb meegemaakt. Het is zonder twijfel ook een van de meest waardevolle uitdagingen. Aan alle mensen die mij hierbij de afgelopen jaren geholpen hebben: dank u!

Cesar Vandeveld  
Kortrijk, december 2016

# CONTENTS

LIST OF ACRONYMS	9
SAMENVATTING	13
SUMMARY	17
I INTRODUCTION	21
1.1 A New Generation of DIY	23
1.1.1 Open Source Software	24
1.1.2 Open Source Hardware	26
1.1.3 Hacking Paradigm	32
1.1.4 Maker Movement	33
1.1.5 DIY in Research	36
1.2 Learning & Creativity	37
1.2.1 STEM Education	37
1.2.2 Robot Kits as Learning Material	42
1.2.3 Creativity	50
1.2.4 Flow	51
1.3 Social Robotics – An Emerging Technology	53
1.3.1 Classification	56
1.3.2 Embodiment and Appearance	60
1.3.3 DIY and Open Source in Robotics Research	62
1.4 Research Question and Design Goals	66
1.5 Outline	68
1.6 List of Publications	71

<b>2</b>	<b>DESIGN METHODOLOGY</b>	<b>73</b>
2.1	User-centered Design	75
2.1.1	Usability & User Experience	79
2.2	Iterative Prototyping	88
2.2.1	Digital Manufacturing techniques	89
2.2.2	Design Strategies	91
2.3	Conclusion	95
<b>3</b>	<b>DESIGN ITERATIONS</b>	<b>97</b>
3.1	Hexapod Robots	98
3.1.1	Stigmergic Ant	98
3.1.2	Locomotion Algorithm	101
3.1.3	Scorpion	105
3.1.4	Summary	107
3.2	Robot Blocks – Toolkit for Simple Educational Robots	108
3.2.1	Robots to Motivate Students into STEM	109
3.2.2	Design of the Building Systems	112
3.2.3	Measuring Usability, Affective Appraisal, and Functionality	118
3.2.4	Results	119
3.2.5	Summary	122
3.3	Ono – Generation 1	124
3.3.1	Conceptual Design of the Embodiment	126
3.3.2	Construction	130
3.3.3	Reproduction of a Robot	132
3.3.4	Robot-Assisted Therapy	136
3.3.5	Summary	141
3.4	Ono – Generation 2	142
3.4.1	Skeleton and Module Improvements	143
3.4.2	Workshop at UNN	146
3.4.3	Workshop at HRI-SS	147
3.4.4	Using Ono in Therapy	152
3.4.5	Summary	154
3.5	Opsoro Platform	155
3.5.1	Design Workshop at TEI	156
3.5.2	The Illusion of Life	161
3.5.3	Classroom of the Future	170
3.5.4	Summary	175
3.6	Conclusion	176

<b>4</b>	<b>OPSORO: OPEN PLATFORM FOR SOCIAL ROBOTICS</b>	<b>177</b>
4.1	Hardware	178
4.1.1	Modules	179
4.1.2	Workshop Base	188
4.1.3	Embodiment	188
4.2	Electronics	194
4.2.1	First Generation – Microcontroller-based	195
4.2.2	Second Generation – Microcomputer-based	196
4.2.3	Sensing Touch	204
4.2.4	Controlling Servos	206
4.3	Software	208
4.3.1	Facial Expression Algorithm	208
4.3.2	User Interface Precursors	215
4.3.3	App-based Web Interface	216
4.4	Conclusion	230
4.4.1	Future Enhancements	230
4.4.2	New Developments	231
<b>5</b>	<b>CONCLUSION &amp; FUTURE PERSPECTIVES</b>	<b>233</b>
5.1	Open Hardware in Robotics Research	237
5.2	Entrepreneurship and Valorization	239
	<b>BIBLIOGRAPHY</b>	<b>243</b>
	<b>LIST OF TABLES</b>	<b>261</b>
	<b>LIST OF FIGURES</b>	<b>267</b>





# LIST OF ACRONYMS

<b>ABS</b>	Acrylonitrile Butadiene Styrene.
<b>AC</b>	Alternating Current.
<b>AI</b>	Artificial Intelligence.
<b>AJAX</b>	Asynchronous Javascript and XML.
<b>API</b>	Application Programming Interface.
<b>ASD</b>	Autism Spectrum Disorder.
<b>AU</b>	Action Unit.
<b>B-Rep</b>	boundary representation.
<b>CAD</b>	Computer Aided Design.
<b>CLK</b>	clock.
<b>CNC</b>	Computer Numeric Control.
<b>CS</b>	Chip Select.
<b>CSG</b>	Constructive Solid Geometry.
<b>CSI</b>	Camera Serial Interface.
<b>CSS</b>	Cascading Style Sheets.
<b>DAC</b>	Digital to Analog Converter.
<b>DC</b>	Direct Current.
<b>DIY</b>	Do-It-Yourself.
<b>DOF</b>	Degree of Freedom.
<b>EVA</b>	Ethylene-Vinyl Acetate.

<b>FACS</b>	Facial Action Coding System.
<b>FDM</b>	Fused Deposition Modeling.
<b>FET</b>	Field-Effect Transistor.
<b>FSR</b>	Force-Sensitive Resistor.
<b>GPIO</b>	General Purpose Input/Output.
<b>GUI</b>	Graphical User Interface.
<b>HAT</b>	Hardware Attached on Top.
<b>HCI</b>	Human-Computer Interaction.
<b>HRI</b>	Human-Robot Interaction.
<b>HTML</b>	Hypertext Markup Language.
<b>HTTP</b>	Hypertext Transfer Protocol.
<b>I<sup>2</sup>C</b>	Inter-Integrated Circuit.
<b>I<sup>2</sup>S</b>	Inter-IC Sound.
<b>IC</b>	Integrated Circuit.
<b>IDE</b>	Integrated Development Environment.
<b>IK</b>	Inverse Kinematics.
<b>IoT</b>	Internet of Things.
<b>IP</b>	Internet Protocol.
<b>IR</b>	infrared.
<b>ISR</b>	Interrupt Service Routine.
<b>LED</b>	Light Emitting Diode.
<b>LIDAR</b>	Light Detection And Ranging.
<b>MDF</b>	Medium-Density Fibreboard.
<b>mDNS</b>	multicast Dynamic Name System.
<b>MISO</b>	Master-In Slave-Out.
<b>MOSFET</b>	Metal-Oxide-Semiconductor Field-Effect Transistor.
<b>MOSI</b>	Master-Out Slave-In.
<b>NURBS</b>	Non-Uniform Rational B-Splines.

<b>OS</b>	Operating System.
<b>OSHW</b>	open source hardware.
<b>PBL</b>	Project-Based Learning.
<b>PCB</b>	Printed Circuit Board.
<b>PLA</b>	Polylactic Acid.
<b>PS</b>	polystyrene.
<b>PU</b>	Polyurethane.
<b>PWM</b>	Pulse-Width Modulation.
<b>RAT</b>	Animal-Assisted Therapy.
<b>RAT</b>	Robot-Assisted Therapy.
<b>RC</b>	Radio-Controlled.
<b>RFID</b>	Radio-frequency identification.
<b>ROS</b>	Robot Operating System.
<b>RUI</b>	Robotic User Interface.
<b>SBC</b>	Single-board Computer.
<b>SMT</b>	Surface-Mount Technology.
<b>SoC</b>	System-on-Chip.
<b>SPI</b>	Serial Peripheral Interface.
<b>STEAM</b>	Science, Technology, Engineering, Arts and Math.
<b>STEM</b>	Science, Technology, Engineering and Math.
<b>SUS</b>	System Usability Scale.
<b>TUI</b>	Tangible User Interface.
<b>TVS</b>	Transient Voltage Suppression.
<b>UART</b>	Universal Asynchronous Receiver/Transmitter.
<b>UCD</b>	User-Centered Design.
<b>UI</b>	User Interface.
<b>URL</b>	Uniform Resource Locator.
<b>USB</b>	Universal Serial Bus.

<b>UX</b>	User Experience.
<b>YAML</b>	YAML Ain't Markup Language.
<b>ZPD</b>	Zone of Proximal Development.

# SAMENVATTING

De voorbije jaren worden gekenmerkt door een verhoogde interesse in robotica, waarbij analisten steevast wijzen naar robotica als één van de volgende grote technologische trends. Dit is niet verrassend; de technologie werd gestaag verbeterd in de afgelopen decennia en bereikt nu uiteindelijk een kantelpunt waarop robots niet langer beperkt zijn tot de fabrieksomgevingen. Geleidelijk verschuift de aandacht binnen robotica van robots die functioneren binnen hun eigen afgebakende ruimtes naar robots die samenleven in de natuurlijke habitat van mensen. Naar gelang het gebruik van robots in het dagelijkse leven steeds vaker voorkomt, wordt het belang van intuïtieve communicatie met deze complexe systemen duidelijk. Simpelweg gezegd: we zouden onze wereld niet moeten aanpassen aan robots. In plaats daarvan moeten nieuwe robots zo ontworpen worden dat ze om kunnen gaan met onze wereld op een zinvolle manier. Deze evolutie zorgt niet alleen voor meer focus op robot veiligheid en conformiteit, het resulteert ook in stijgende interesse in sociale robotica en Human-Robot Interaction (HRI), een onderzoeksveld dat betrekking heeft op de natuurlijke interactie tussen robots en mensen. Sociale robots zijn robots die kunnen communiceren door middel van intuïtieve sociale vaardigheden, zoals bijvoorbeeld door middel van emoties, gelaatsexpressies, spraak, en oogcontact. Echter, onderzoek binnen mens-robot interactie houdt zich momenteel voornamelijk bezig met het programmeren van gedragingen op standaard robots, en het ontwerp van de belichaming van de robots blijft vaak een onverkende ontwerpdimensie. Het snel kunnen ontwerpen van belichamingen voor sociale robots heeft een groot potentieel. Eén van de doelen van dit werk is om het ontwerp van belichamingen voor sociale robots te vergemakkelijken, om zo het verborgen potentieel van deze ontwerpdimensie naar voor te brengen.

Momenteel zien we een andere grote trend die de moderne maatschappij zal beïnvloeden: de heropleving van het do-it-yourself (DIY) paradigma. De vernieuwde focus van DIY gaat veel verder dan de amateur-radio's en huis-tuin-en-keuken klusjes van weleer. De hedendaagse DIY trends, zoals de maker movement en de open source hardware beweging, omarmen en belichamen technologie in zijn geheel. Deze trends zijn verantwoordelijk voor het toegankelijk maken van heel wat exclusieve en complexe technologieën, zoals bijvoorbeeld microcontrollers, 3D printers, CNC machines en robots. Het belang hiervan is niet de technologie op zich, maar het feit dat deze technologieën toegankelijk gemaakt worden voor een breed publiek, wat leidt tot nieuwe en onverwachte toepassingen.

Het werk dat beschreven wordt in dit proefschrift bevindt zich op het kruispunt tussen

deze twee grote trends. Het proefschrift bespreekt de oorsprong en de ontwikkeling van Opsoro (open platform for social robots), een DIY platform dat amateurs en non-experts in staat stelt om zelf sociale robots te ontwerpen en te bouwen. Het platform werd specifiek ontworpen om open source, goedkoop, en gemakkelijk in gebruik te zijn. De platform-aanpak wordt gebruikt om sociale robotica technologie toegankelijk te maken voor een breed publiek. Vaak zijn robot-ontwerpers en robot-gebruikers twee afzonderlijke groepen, elk met hun eigen kennis en vaardigheden. Het is niet vanzelfsprekend om kennis uit te wisselen tussen deze partijen. Daarom is het zinvol om gebruikers in staat te stellen om zelf hun robots te bouwen in plaats van een oplossing voor hen te ontwerpen.

Het ontwerp van een platform is een veel complexer gebeuren dan het ontwerp van een product. Een product wordt door één persoon ontworpen en wordt door een ander persoon gebruikt. Een platform wordt ontworpen door een systeemontwerper, het wordt gebruikt door een tussenpersoon om een product te ontwerpen, en dat product wordt uiteindelijk gebruikt door een eindgebruiker. Het ontwerp van een robotica-platform is niet alleen uitdagend vanuit een technisch standpunt, maar ook vanwege de impact van menselijke factoren op alle aspecten van het systeem. Bijgevolg schieten traditionele engineering-methodieken tekort, en moet er een andere aanpak worden gebruikt. In dit werk wordt daarom een holistische, user-centered ontwerpmethodologie gehanteerd. Deze aanpak houdt rekening met de rol van verschillende gebruikers doorheen het ontwerpproces, en steunt op meerdere iteraties om stapsgewijs inzichten te verwerven in de gebruikseisen en de gebruikcontexten. Binnen dit proces wordt het belang van gebruiksgemak en gebruikservaring benadrukt omdat deze aspecten de sleutel vormen tot het slagen van complexe, iteratieve systemen. Doorheen dit werk wordt gebruik gemaakt van digitale productietechnieken zoals 3D printing en laser cutting om het ontwerpproces te ondersteunen. De ontwerpbeslissing om digitale productietechnieken te gebruiken heeft implicaties op de doelstellingen van dit project. Binnen digitale productietechnieken wordt de informatie die nodig is om een onderdeel te produceren verrat binnen een digitaal bestand. Computergestuurde machines gebruiken deze informatie om automatisch fysieke onderdelen te creëren zonder een beroep te doen op de vaardigheid of het vakmanschap van de operator van de machine. Dit laatste punt is belangrijk voor de reproduceerbaarheid van een ontwerp: het vergemakkelijkt delen via het internet, het verlaagt de barrière om een ontwerp te kopiëren, en het zorgt voor een hoge graad aan herhaalbaarheid. Dit heeft een belangrijke invloed op de openheid van het systeem, en maakt het mogelijk om online communities te stichten. Een andere intrinsieke eigenschap van digitale productietechniek is dat een hoge graad van ontwerpcomplexiteit gebruikt kan worden, waardoor ontwerpers extra functionaliteiten (bv. instructies) kunnen integreren in de vormgeving van een onderdeel. Dit aspect wordt gebruikt om het gemakkelijker te maken om ontwerpen te reproduceren en te assembleren.

De vele ontwerpiteraties hebben uiteindelijk geleid tot de huidige versie van het Opsoro platform. Iedere iteratie zorgde voor andere inzichten in het slagen of falen van verschillende aanpakken, en binnen de verschillende iteraties werden de *ontwerp*-, *bouw*-, en *gebruik*-fasen van een platform verkend. Het is belangrijk om aan te geven dat het ontwerpproces niet lineair is. Het is een complex netwerk van ontwerpen die geïnspireerd zijn op en beïnvloed zijn door andere ontwerpen. Binnen dit proefschrift worden vijf ver-

schillende projecten beschreven. Het eerste project gaat over wandelende hexapod robots. Dit was één van de eerste exploraties binnen DIY robotica, en beïnvloedde de constructietechnieken die gebruikt werden doorheen de rest van dit werk. Ten tweede wordt het ontwerp van een bouwkit voor kleinschalige educatieve robots beschreven. Het derde en vierde project gaat over het ontwerp van Ono, een DIY sociale robot gemaakt voor interactie-experimenten met kinderen. Tenslotte gaat het vijfde project over het Opsoro platform, wat verder bouwt op de voorgaande projecten. Binnen elk van deze projecten werd experimenten met amateurgebruikers uitgevoerd waarin de ontwerp-, bouw-, en gebruiksfases van de tools telkens onderzocht werden.

Het Opsoro platform wordt omschreven als het finale project binnen dit werk. Echter, het ontwerpproces wordt verder gezet en het systeem wordt momenteel verder ontwikkeld met oog op commercialisatie. Er wordt momenteel gewerkt om tegen 2018 een Opsoro starter kit te lanceren als product, met een sterke focus op STEAM-educatie. Het nieuwe platform bouwt verder op de stevige fundamenteën van dit proefschrift om te leiden tot een uitbreidbare set van modulaire maker-kits die kunnen gebruikt worden om eigen sociale robots te creëren en te bouwen. Verder wordt de software momenteel omgevormd tot een online ontwikkelingsomgeving en educatieve community. De web-gebaseerde interface stelt gebruikers in staat om hun creaties te bedienen via een groeiend aantal apps, wat leren en speelactiviteiten op maat van iedere gebruiker mogelijk maakt. Hiermee hopen we de kennis die verzameld werd binnen dit onderzoek toegankelijk te maken voor het brede publiek, om zo een significante impact te hebben op de manier waarop sociale robots ontworpen worden door de toekomstige generatie.





# SUMMARY

Recent years have been characterized by a heightened interest in robotics, and analysts have steadily pointed at robotics as one of the next big trends in technology. This is perhaps unsurprising; the technology has steadily improved over the past decennia and is now reaching a tipping point where robots are no longer constrained to factory environments. Gradually, robotics research has shifted its attention from robots that function within their own predefined space to robots that coexist with humans in the human's natural habitats. As robotic agents become more and more commonplace, the importance of intuitive communication with these complex systems becomes apparent. Simply put: we should not need to adapt our world to suit robots. Instead, new robots should be designed so that they can interact with our world in a meaningful way. This evolution has not only driven interest in robot safety and compliance, it has also resulted in the study of social robotics, a field that is concerned with natural interaction between robots and humans. Social robots are robots that can communicate using social affordances that we find intuitive, for example through emotions, facial expressions, speech, and gaze. However, research within Human-Robot Interaction (HRI) is currently mostly concerned with programming robot behaviors on standard robots, whereas the embodiment design is frequently left as an unexplored design dimension. The rapid design of new social robotic embodiments holds great potential. One of the goals of this work is to facilitate the embodiment design of social robots, unlocking the hidden potential of this design dimension.

At the same time, another big trend is shaping and disrupting modern society: the revival of the do-it-yourself (DIY) paradigm. This renewed focus on DIY moves beyond ham radios and home improvement projects of old. Contemporary DIY trends, such as the maker movement and the open source hardware movement, embrace and embody technology in its entirety. These trends have been responsible for democratizing previously exclusive and complex technology, including microcontrollers, 3D printers, CNC machines and robots. The importance here is not the technology per se, but rather that these technologies have been made accessible outside of a very small community of experts, leading to novel and unexpected applications.

The work described in this dissertation is situated at the intersection of these two trends. This dissertation discusses the inception and development of Opsoro (open platform for social robots), a DIY platform that enables amateurs and non-experts to design and create custom social robots. The platform is specifically designed to be open source, low cost, and

easy to use. The platform approach is used in order to make social robotics technology accessible to a wider audience. Often, robot designers and robot users are two distinct groups, each with their own skills and knowledge, and it is not always easy to transfer this knowledge. Therefore, it makes sense to give users the tools to build their own solutions instead of designing a solution for them.

Designing a platform is much more complex than designing a product. A product is designed by one person and used by another. A platform is designed by a system designer, it is used by an intermediary to create their own product, and that product is finally used by an end user. The design of a robot platform is challenging not only from a technical point of view, but also because of the impact of the human factors on all aspects of the system. Accordingly, traditional engineering methodologies fall short and a different approach must be used. In this work, a holistic user-centered design approach is used instead. This approach acknowledges the role of different users throughout the design process, and relies on multiple iterations to gradually gain an understanding of user requirements and usage contexts. Within this process, the methodology emphasizes usability and user experience aspects as they hold the key to success for complex, interactive systems. Throughout this work, digital manufacturing technologies such as 3D printing and laser cutting were used to support the design process. The design decision to use digital fabrication has implications for a number of the project goals. With digital fabrication techniques, the information required to produce a part is contained within a digital file. Computer-controlled machines use this data to produce physical parts, requiring little skill or artisanship from the operator. This last point is important with respect to reproducibility: it facilitates online sharing, it lowers the barrier to making copies and offers a higher degree of repeatability. This directly impacts the openness of the system, as well as the opportunities for online community building. Another intrinsic property of digital fabrication is that design complexity is (nearly) free, affording designers the chance to embed extra functionality (e.g. instructions) in the geometry of a part. This aspect can be leveraged to make the designs easier to reproduce and easier to build.

The design iterations have finally resulted in the current version of the Opsoro platform. Each iteration has led to insights into the success and failure of different approaches, and the different iterations have explored the *design*, *build*, and *use* phases of a platform. More importantly, the design process is not linear. Instead, it is a complex network of designs that have inspired and influenced other designs. The design iterations of five different projects are described in this work. The first project is concerned with the design of walking hexapod robots. This was one of the earliest forays into DIY robotics and informed many of the construction techniques used throughout the rest of this work. Secondly, the design of a construction toolkit for small-scale educational robots is described. The third and fourth projects detail the design of Ono, a DIY social robot created for interaction experiments with children. The fifth and final project is the Opsoro platform, which builds upon the knowledge gained from the preceding projects. Each of these projects has been tested with amateur users in experiments that cover the design, build, and use phases of the tools.

While this work describes the Opsoro platform as the final project, the design process con-

tinues and is currently being developed with commercialization in mind. With a strong focus on STEAM learning, the goal is to launch an Opsoro starter kit product by 2018. The platform builds upon the solid foundations of this dissertation, resulting in an expandable set of modular maker-kits that can be used to create and build your own custom social robot. Furthermore, the software is currently being transformed into an online development environment and educational community. The web-based interface allows users to control their creations through a growing number of apps, enabling learn and play activities tailored to each individual user. With this, we hope to transfer the knowledge gained throughout the research into society at large, significantly impacting the way that social robots can be designed by the future generation.



# Chapter I

## INTRODUCTION

Recent years have been characterized by technology permeating into every aspect of our day-to-day lives. Similarly, we see that the digital world and physical world are converging more each day. One of these recent developments is the emergence of robotic systems in the living environment of humans. Many of these technologies are not even new, they have simply become cheap enough and sophisticated enough for widespread applications. Nevertheless, by becoming so ubiquitous they impact society tremendously, forcing us to rethink many well-established concepts.

On the other hand, we see a paradox emerge in that the proliferation of technology goes hand in hand with technology disappearing from our sight. Indeed, researchers in human-computer interaction have long since argued that interfaces should move beyond screens and buttons as our window into the digital world. For instance, Ishii and Ullmer (1997) have advocated the direct manipulation of digital information through tangible artifacts. Weiser (1994) posits that “a good tool is an invisible tool”, indicating that technology should be subjugate to the action; it is simply a means to an end. Finally, in his book, User Experience (UX) designer Krishna (2015) simply argues that “the best interface is no interface”.

These perspectives of the future are slowly, yet steadily becoming reality. Cars unlock themselves when they detect the proximity of a key fob. Nest thermostats observe our habits over time and learn to adjust the room temperature automatically. On phones, Google Now presents personalized information, such as airplane boarding passes, or package tracking, right when the user needs it, *without* the user requesting it explicitly. What all these products have in common is that they tend to stay in the background, putting as few steps as possible between ourselves and our goals.

One of the many ways in which these trends are expressed is the emergence of social robotics, the domain in which this treatise is situated. Robots have been used in factories for well over 50 years. Now, more and more applications are starting to emerge outside of

the manufacturing niche.

As robotic agents become more and more commonplace, the importance of intuitive communication with these complex systems becomes apparent. Robotic agents may also evolve into the role of a mediator between humans and artificial systems, providing an interface to these systems through natural and intuitive communication. This is one of many ways in which human-computer interaction can evolve beyond an unintuitive reliance on screens and buttons. For us humans, intuitive interaction includes things like verbal communication, gestures, and facial expressions. After all, it is with these tools that we interact among ourselves.

Previous work has already proven the effects of embodiment design on how people perceive robots (Wainer et al., 2006), as well as the benefits of embodied agents over virtual agents (Lee et al., 2006; K. Williams and Breazeal, 2013). However, research within Human-Robot Interaction (HRI) currently mostly concerned programming robot behavior on standard robots (e.g. Nao), whereas the embodiment design is frequently left as an unexplored design dimension. The rapid design of new social robotic embodiments holds great potential. One of the goals of this work is to facilitate the embodiment design of social robots, unlocking the hidden potential of this design dimension.

The work described in this thesis also plays into a second aspect of technology proliferation, namely the aspect of the resurgence of Do-It-Yourself (DIY) in the form of the maker movement, a subculture that is characterized by the democratization of technology in a creative, hands-on setting. The movement draws attention to the empowering feeling of making things, noting that we should not restrict ourselves to being just consumers of technology, but that we should also become creators of technology in order to shape the world around us.

Characteristic examples of previously exclusive, complex technology that has been made accessible to a larger public through this movement includes microcontrollers, 3D printers, CNC machines, and robots. Many engineering challenges that were considered extremely difficult in the past can now be accomplished with the help of open designs and off-the-shelf parts. For instance, building a quadcopter from scratch used to require in-depth knowledge of control theory, brushless DC motor control, software engineering, composite material design, etc. Yet nowadays, these devices can be built by hobbyists through the creative recombination of existing designs. This shift is largely due to the work of open source communities. The novelty of these projects is not in the technology per se, but in the fact that they are made accessible outside of a very small community of experts, leading to novel and unexpected applications.

There are many different circumstances that have lead to this renaissance of DIY. One could argue that its emergence was simply the inevitable result of the rapid evolution of technology, though notable influences include the rise of the internet as a communication medium, the open source (software) movement, the steadily continuously costs of electronics, and the rise of FabLabs. Well-known technologies that embody this paradigm include the Arduino microcontroller platform and the RepRap low-cost 3D printers.

The evolution of this DIY culture in the past few years has opened exciting new opportunities in many different areas of art and technology, ranging from low-cost prostheses, to hackable scientific instrumentation, to new kinds of manufacturing machines. With the work presented in this thesis, we introduce aspects of the DIY culture within the field of human-robot interaction, illustrating the merits of putting social robot technology in the hands of amateurs. Our efforts have culminated in the design of a DIY platform, named Opsoro, that enables amateurs to design custom social robots from scratch. We hope that this platform will enable and inspire a new generation of human-robot interaction applications.

## I.1 A NEW GENERATION OF DIY

DIY, short for Do-It-Yourself, is an umbrella term for activities where amateurs engage in the design, modification, or repair of artifacts. The term “amateur” does not necessarily mean that DIYers are unskilled, but rather that the activities are performed outside of a professional capacity (Kuznetsov and Paulos, 2010). Wolf and McQuitty (2011) define DIY as “activities in which individuals engage raw and semi-raw materials and component parts to produce, transform, or reconstruct material possessions, including those drawn from the natural environment”. While economic factors do play a part in the motivation for participating in DIY activities (Wolf and McQuitty, 2011), monetary gains are usually not the sole motive of DIYers. DIYers also frequently cite reasons such as creative expression, the desire to learn new skills, and the need for objects that do not exist commercially (Kuznetsov and Paulos, 2010). Furthermore, DIY projects often give the creator a sense of accomplishment and fulfillment. This sense of accomplishment forms the basis for the so-called “IKEA effect”, meaning that consumers place increased value in products that they (partially) made themselves (Norton et al., 2012).

The act of creating objects is as old as humanity itself. One of the defining traits of our species is our reliance on tools to survive and thrive. From as early as prehistory, people have felt the need to create tools and devices to manipulate the environment or to help them perform certain tasks. Over time however, individuals have adopted increasingly specialized roles in their communities. Roles in hunter-gatherer societies were much more generalized, whereas the subsequent agrarian revolution and industrial revolution have given way to much more specialized roles. Owing to the economies of scale, this specialization results in increased productivity, giving individuals more and more time to pursue activities that are not directly related to survival and leading to advances that otherwise would not have been possible. However, increased specialization is paired with decreasing self-reliance, as individuals are no longer capable of survival without the aid of others. DIY culture is reactionary in this regard, and self-sufficiency is a commonly reoccurring motivation in DIY trends.

DIY is a trend that reoccurs every few decades. Typically, each new incarnation of the DIY trend is triggered by societal or technological advances, and is characterized by a certain degree of rebelliousness. Some examples of DIY movements in the 20<sup>th</sup> century include:

- *The Arts & Crafts movement* (ca. 1880 - 1910) – an anti-industrial movement advocating traditional craftsmanship in design and decoration (Naylor, 1980).
- *Ham radio* (ca. 1920s) – an electronics hobby where amateurs build and operate radio communications devices. The hobby continued to thrive during WW II, despite a ban on amateur radio communication (Haring, 2003; Massie and Perry, 2002).
- *Home improvement* (ca. 1970s) – a trend that saw a peak in popularity in the decades post-WW II, influenced by the social and environmental views of the 1960s. Even though its popularity has partially subsided, DIY home improvement is still practiced by many today (Goldstein, 1998).
- *Homebrew computing* (ca. 1970s) – A movement centered around the Homebrew Computer Club in Silicon Valley, focused on building computers from scratch. The homebrew computing movement helped enable the personal computer revolution with devices such as the Apple I (Ceruzzi, 2003; Dougherty, 2008).

The past ten years have been marked by a renewed interest in DIY culture. This contemporary form of DIY is commonly referred to as the *maker movement*. A key characteristic of modern DIY is that it takes place at the interface between the physical world and the digital world. To elaborate, the projects typically incorporate significant amounts of software design as well as hardware design, with both aspects equally crucial to functionality. Frequently, these artifacts can be described as mechatronic in nature, incorporating both mechanics and electronics. To illustrate, the design of a 3D printer – a common Maker-centric project – requires mechanical engineering for the printer’s frame and motion systems, electrical engineering for the device’s driver electronics, as well as software engineering to create the firmware and host software. Without any one of these three, the artifact would become useless.

The second important characteristic is the significant role that the internet plays in collaboration. Online platforms such as Instructables and Github enable design collaboration and sharing information. Digital design files of physical objects are then transported to the physical world using digital manufacturing devices, such as laser cutters or 3D printers.

The next sections will further elaborate on contemporary DIY practices, discussing open source software & hardware, hacking, and the maker movement. One should note that there is significant overlap between these trends, and that the difference between them is not always clear-cut. This is compounded by the informal, loose-knit structure of DIY communities. Consequently, some terminology will be used interchangeably throughout the rest of this work.

### 1.1.1 OPEN SOURCE SOFTWARE

The term “open source” refers to the act of making the source files of a certain design available for anyone to view, modify, and use. This idea originates from the software



world, where programmers would make the source code of their programs available.

Most software is typically distributed as a binary. Transforming human-readable source code into a binary is simple, and is done automatically by a compiler. Conversely, turning a binary back into source code is nearly impossible, and requires extensive reverse engineering. Depending on the complexity of the software, such a process is prohibitively difficult and time consuming. Consequently, for practical purposes, access to the source code is required in order to make meaningful changes to an existing piece of software.

In early computing history, sharing source code was the norm, not the exception. Most programs were custom created by users to fulfill a specific purpose, and code from others was often used to bootstrap development of new programs. Most computer users in the 1960s and 1970s were part of academic or corporate research labs, and saw the free exchange of source code as a normal part of research culture. Only at a later stage did this pattern change. With the rise of commercial, general-purpose software, most applications were distributed as binaries without access to the source code (Hippel and Krogh, 2003).

This mentality change later galvanized the emergence of modern open source software communities. These communities, comprised of hobbyists as well as professional programmers working in their spare time, created some of the world's most widely used software, including the Linux operating system and the Apache web server. The internet proved an important factor in fostering these communities: it enabled collaboration on a much larger scale than simply through sharing floppy disks with lab colleagues, as was done before.

In “*The cathedral and the bazaar*”, Raymond (1999) describes the zeitgeist of open source communities during the 90's. The author describes two different development models of open source communities: the top-down *cathedral model* and the bottom-up *bazaar model*. The author likens the development process of important and complicated software (such as an Operating System (OS)) to cathedral-building, relying on a very small group of “wizards” to do the initial development. The opposite approach is dubbed the bazaar model by the author. It is characterized by many different agendas and approaches, as opposed to the singular expert-directed focus of the cathedral model. The author then describes how the bazaar model led to the success of Linux, a complex project where one would expect that the cathedral model should be used. This bazaar model was intentionally used by the author during the development of fetchmail, and the paper summarizes lessons learned from applying the bazaar approach in practice.

Some large software companies, such as Google and Red Hat, have embraced the open source software movement as part of their business model. Not only do they build their products upon open source software, they also release parts of their own products under an open source license. Eastham (2015) discusses this phenomenon, noting that this is not necessarily pure altruism: an open source strategy can be used to bolster their own products by making them better, safer, and more widespread. Further analysis on the motivations for open source software is done by Hippel and Krogh (2003), who purposes a hybrid form between the “*private investment*” model and the “*collective action*” model as a theoretical framework for the open source phenomenon.

Over time, open source evolved beyond the domain of software, resulting in a more generalized open source movement. The core idea behind open source software was adapted to suit different areas of technology. In most areas, there is an analogous concept to the source code files of open source software, where form *A* is easily turned into form *B*, but the reverse direction is extremely difficult. Thus, “source” is interpreted in a broader way, and can encompass things such as recipes, blueprints, models, instructions, etc.

One of the more esoteric examples of this principle is OpenCola<sup>1</sup>, a soft drink similar to Coca Cola of which the recipe was released under the GNU General Public License. Similarly in 2016, the Scottish brewery BrewDog released its recipe collection under an open source license (Mason, 2016). One final example is *Open Design Now* (Abel et al., 2011), a book whose chapters were gradually made available on the internet under a Creative Commons license.

## 1.1.2 OPEN SOURCE HARDWARE

The previous section illustrated the evolution of open source software in the late 90s and early 2000s, concluding with examples of the use of the open source model outside of software. This section will discuss this point further, elaborating on one of the most influential trends within the open source community in recent years: open source hardware (OSHW) (Powell, 2012). As the name implies, open source hardware is a physical artifact of which the original design files have been made available, and anyone is free to reproduce and modify the artifact. There are many different definitions of OSHW, and each has its own nuances. The TAPR open hardware license, for instance, offers the following definition (TAPR, 2007):

*“Open Hardware is a thing – a physical artifact, either electrical or mechanical – whose design information is available to, and usable by, the public in a way that allows anyone to make, modify, distribute, and use that thing.”*

While the idea behind OSHW is not entirely new (Gibb, 2014, p. 11), two factors have been pivotal for the interest and wider adoption of the movement in recent years:

1. Access to advanced manufacturing processes, such as 3D printing, laser cutting, and Printed Circuit Board (PCB) manufacture has improved drastically. Low-cost 3D printers, FabLabs, and online manufacturing services have opened the door to personal manufacturing (D. A. Mellis, 2014).
2. The emergence of new sharing mechanisms (e.g. forums, instructions, video, images), which plays a major role in motivating and sustaining communities of builders, crafters and makers (Kuznetsov and Paulos, 2010).

<sup>1</sup> Soft Drink Formula Version 1.1.3 –  
[http://alfredo.octavio.net/soft\\_drink\\_formula.pdf](http://alfredo.octavio.net/soft_drink_formula.pdf)

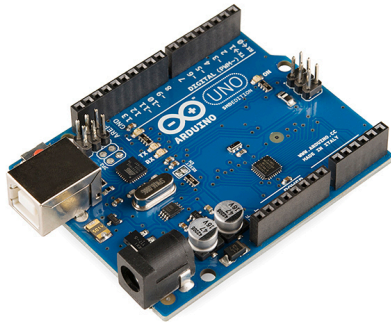
Open source hardware has a wide and varied audience, including amateurs and hobbyists, commercial companies, and researchers. Each of these groups has their own motivation to engage in OSHW activities. Amateurs report reasons such as receiving feedback, educating others, and showcasing their work (Kuznetsov and Paulos, 2010). For companies, motivations can include reducing R&D costs, incorporating community developments, as well as creating a platform around the product (Gibb, 2014, p. 216). Finally, A. Williams et al. (2012) draw a parallel between the peer review process in research and the process of open sourcing research hardware, noting that doing so increases the legitimacy of the work because others can inspect it and contribute to it.

Open-sourcing hardware is not a black-and-white matter, but rather a continuous spectrum of varying degrees of openness (Gibb, 2014; Yanamandram and Panchal, 2014). The effect is also much more noticeable than with open source software. At the most rudimentary level, it may simply mean that the creators provide a pinout diagram or basic maintenance instructions. Another nuance is in the design toolchain: many OSHW projects use proprietary design tools such as EAGLE or SolidWorks to create their design. Some OSHW practitioners strive for a completely open workflow, opting for open source design software wherever possible. Gibb (2014, p. 253) notes that open hardware will probably never completely cover all layers of hardware. In extremis, the process would have to extend down to the level of the production of raw materials, which seems unlikely.

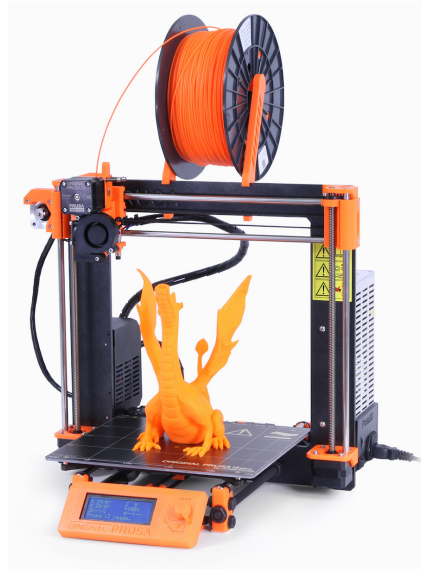
One of the first large-scale success stories of OSHW is the Arduino microcontroller platform (fig. 1.1) (D. A. Mellis, Igoe, et al., 2007). The platform consists of a physical hardware PCB, as well as an Integrated Development Environment (IDE) application and a set of software libraries to program the PCB. Arduino was originally designed to be used by students at the Interaction Design Institute Ivrea, focusing on enabling artists and designers to play with electronics.

While the Arduino board itself is admittedly fairly simple, the project itself significantly influenced contemporary DIY and OSHW culture. Arduino was also one of the first commercially successful OSHW products (Thompson, 2008). All relevant software and hardware source files were released as open source, but the name “Arduino” was registered as a trademark by the company (Thompson, 2008). Consequently, others were free to copy and modify the board as they please, but they could not call their board an Arduino. Still, this did not deter others from creating variants (D. Mellis and Buechley, 2012). The Arduino business model was later copied by a number of other OSHW companies.

Another prominent example is the RepRap project (fig. 1.2) (Jones et al., 2011), a project with the aim of creating a self-replicating Fused Deposition Modeling (FDM) 3D printer. The project was influential in making 3D printing available to a wider audience, and the large majority of the sub 2000 EUR 3D printers currently on the market can trace their lineage back to the RepRap project. RepRap 3D printers are complex mechatronic machines, and the hardware goes well beyond the simple PCB of the Arduino project. As Yanamandram and Panchal (2014, p. 106) illustrates through a product hierarchy diagram, RepRaps have many intricate hardware subsystems, including the control electronics, the 3-axis cartesian motion system, the extruder nozzle assembly and the extruder feed mechanism. On top of that, they also require a significant body of software to function,



**Fig. I.1** Arduino Uno<sup>3</sup>



**Fig. I.2** RepRap Prusa Mendel i3<sup>4</sup>

including firmware, model slicing software, and host software. Despite the complexity, a RepRap can be built with a completely open stack. This has led to an explosive growth of RepRap-derivative 3D printers, as evidenced by the RepRap family tree<sup>2</sup>.

Open source hardware is a very recent phenomenon. While the OSHW approach has seen some early successes, many obstacles remain. In the remainder of this section, we will discuss the following challenges:

1. Cost of replication
2. Documentation
3. Design software
4. Licensing & commercialization

One of the ambitions of open source is to allow anyone to build and modify a certain work. In the past, this has worked very well in software. However, while duplication of digital data is practically free, there is a measurable cost associated with replicating hardware. In *The cathedral and the bazaar* Raymond (1999) lists a number of properties of successful open source software projects. Releasing code early and often is one of the key points, as is outsourcing as much work as possible to members of the community.

<sup>3</sup>Photo by Sparkfun Electronics. Licensed under Creative Commons Attribution 2.0.

<sup>4</sup>Photo by Josef Prusa. Licensed under GNU Free Documentation License 1.2.

<sup>2</sup>RepRap Family Tree – [http://reprap.org/wiki/RepRap\\_Family\\_Tree](http://reprap.org/wiki/RepRap_Family_Tree)

One difference with software is the *barrier* to simply copy a work, a prerequisite to collaboration. This barrier encompasses skill, time, and component costs. While digital manufacturing techniques have dramatically lowered the skill barrier to produce components, other skills are required to come to a meaningful artifact, such as assembling and soldering. D. Mellis and Buechley (2012) note that while many derivative Arduino boards have been created, few hardware changes have made their way back to the Arduino project. D. A. Mellis (2014, p. 154) notes that developers may be inclined to recuperate some of their costs by selling a derivative instead of contributing changes back to the original author.

Several practitioners identify modularity as a key aspect for the success of open source hardware. Yanamandram and Panchal (2014) notes the following: “One of the biggest factors aiding the success of open source software is its capability for modularity and the adequate focus given to the various modules by programmers thereafter. [...] It is necessary to promote particular parts of a system and make them the focus of the development process so that different people can pursue different parts based on their domain knowledge and skillset. This would shorten the learning curve, improve production time and quality.” The same sentiment is echoed by Baafi (2014): One should not have to reinvent the wheel, but rather “pick a tire, pick a spoke, and pick a hub!”. Modularity and interchangeability are one of the key aspects that enable collaboration and darwinian evolution in open hardware. While there certainly are parallels with Raymond’s model of development, open source hardware is separate in its own right. The question remains: does the bazaar model work in open source hardware?

Creating successful open source hardware necessitates more than merely making CAD files available. Not only is it key to design with replication by others in mind, good documentation is essential in order to stimulate broader use and adaptation. Yanamandram and Panchal (2014) argues that the documentation needs of open source hardware go beyond what is normally offered by open source software platforms such as GitHub. Not only do open hardware projects need to manage design files, but also assembly instructions, material and tool documentation, and manufacture procedures. The shape of a physical component can be captured relatively well in a digital file, such as an STL file for 3D printing. However, Kuznetsov and Paulos (2010) show that capturing physical processes is much harder. Tseng and Resnick (2014) shows that many open source hardware practitioners experience *designing* and *documenting* as two distinct modes. Documenting can be easily forgotten in the process of designing, and documenting while designing can interrupt the flow of the design process. Tseng and Resnick (2014) note that some choose to completely disassemble a project at the end of the design process, so that it can be reassembled and documented step-by-step. However, this method is only possible for small projects that can be rebuilt easily. For larger and more complex designs, this quickly becomes infeasible.

The availability of appropriate design tools that are low-cost, free, or even open source is an important prerequisite for creating open source hardware. Whereas an open source software project might require a compiler to go from source code to an executable binary, open source hardware projects rely on CAD tools to describe a physical object in a digital file. We see software tools as one of the limiting factors for open hardware projects. The idea of

open source hinges upon collaboration between volunteers. More often, this collaboration takes place over the internet. Consequently, tools to modify the design should be easily accessible in order for a project to be successful. Unfortunately, traditional CAD software for mechanical design is prohibitively expensive for amateurs, hampering this collaboration. To illustrate, many designs within the RepRap project are created using OpenSCAD. This open source tool generates polygonal 3D models from a script file describing Constructive Solid Geometry (CSG) operations. OpenSCAD allows for parametric designs, so that changing one variable (e.g. the size of the build volume) would automatically cause all appropriate components to be regenerated. Though powerful and free, the system is complex to use.

More recently, new mechanical CAD tools have become available, though mostly freeware offerings from commercial companies as opposed to open source software. For instance, Autodesk now offers a number of free-of-charge, easy to use CAD applications under the 123D moniker. These tools include a mesh editor, a photogrammetry tool, and a CSG tool for mechanical designs. In addition to these maker-centric tools, they also offer a cloud-based traditional mechanical CAD suite, free for non-commercial use. In addition, open source software offerings, such as FreeCAD and OpenSCAD, have made great strides in terms of functionality and stability. This is evidenced by projects such as GummiArm (Stoelen et al., 2016), a 3D-printed robot arm that was completely designed in FreeCAD.

Still, many open source hardware projects – especially complex ones – are designed in SolidWorks, a popular commercial CAD package. Examples include OpenHand (Ma et al., 2013), Poppy (Lapeyre et al., 2014), Oncilla (Sproewitz et al., 2011), OpenPCR, Ultimaker 2, and WoodenHaptics (Forsslund et al., 2015). We see a number of potential explanations for this phenomenon. Firstly, it could be due to CAD tool “inertia”, where current projects will continue to use the software they are using, and simply not enough time has passed for new projects to be started using new, free software. A second explanation would be that many projects originate from universities and companies, where researchers have access to expensive CAD software. A third reason could be that free packages that are most in line with traditional mechanical CAD (i.e. Fusion360 and OnShape) are cloud-based, meaning that files are restricted to a 3rd party server, and cannot be synchronized to online repositories. Cloud-based tools also have the downside of making a project dependent on an external company, which could prove problematic should the company ever decide to change, stop, or charge for their product. In any case, appropriate design tools for open source hardware projects are rapidly evolving, and will most likely continue to do so. It will be interesting to see the impact of these new tools on open source hardware in general.

Open source hardware projects originate from many different sources: from amateurs, from academia, from commercial companies. Projects that find their origins in academia typically tend to transition to spinoff companies: it can be difficult to focus on product development (as opposed to pure research) within the framework of a university. Arduino (Interaction Design Institute Ivrea, D. A. Mellis, Igoe, et al. 2007), RepRap (University of Bath, Jones et al. 2011), and Makey Makey (MIT, J. Silver et al. 2012) all originate from academia and later transitioned to independent companies.

Thompson (2008) describes some of the reasons why Arduino chose to remain open source as a company:

- Idealism: open sourcing stimulates copies, thereby spreading the product and leading to wider adoption.
- An open Arduino inspires more interest and more free publicity than it would have if it were proprietary.
- It enables enthusiasts to hack the board. Improvements from the community can be incorporated into the product.

As a company, the challenge is how to be open, but still generate a profit. Intuitively, one would think that giving all design files away for free would be detrimental to business. After all, you have taken the risk to develop a product, and you are inviting other companies to copy you. Thompson (2008) identifies two strategies for OSHW companies:

1. Don't worry about selling a product, sell your expertise instead. As the original creator, the product's community will naturally develop grow around you, giving you an edge on knowledge.
2. Stay ahead of the competition by innovating faster and by offering a higher-quality product. Thompson mentions: "Merely having the specs for a product doesn't mean a copycat will make a quality item."

Still, commercial OSHW is still a very new business model and companies are still experimenting with different methods to stay profitable. For instance, all Arduino designs are open, but the name *Arduino* itself is protected. As Zimmermann (2014, p. 213) remarks: "Brands in open source hardware are as important as they are for businesses with closed source, or patented, hardware." The OSHW approach of Arduino, protecting their brand name via a trademark – while somewhat controversial at the time (D. A. Mellis, 2014, p. 154) – has proven quite successful, and has been copied by other companies.

Makerbot Industries – one of the poster child companies within the 3D printing world – originally started as an open hardware company, and enjoyed considerable amounts of success. In 2012 they opted to transition to a closed, proprietary business model, citing pressure from Chinese clones (Pettis, 2012). The decision led to considerable criticism from the community. Since then, the company was bought by industry giant Stratasys, but faced disappointing sales figures, leading to a layoff of 20% of its staff (Biggs, 2015). Makerbot's competitor, Ultimaker, has remained open source. However, the release of source files is delayed by several months, allowing the company time to recuperate their investments.

As a final point of nuance, it should be noted that hardware is "open" by default (Gibb, 2014, p. 12). Unlike software, which falls under copyright, inventions can only be protected using (costly) patents. Unless patented, anyone is free to reverse engineer an existing

product and bring it to market. The legal framework of open source hardware licenses is more complex than meets the eye. Open source hardware licenses are built upon copyright law. Consequently, such licenses cover only the reference designs, and not the object itself (Thompson, 2008).

### 1.1.3 HACKING PARADIGM

To the layperson the word “hacking” usually invokes the mental image of a criminal breaking into a computer system with the purpose of stealing data or interrupting service. However, the term has a much richer meaning and history beyond its cybercriminality connotation. Older definitions of *hacking* do not have the connotation of malicious intent, and instead focus on cleverness, playfulness and technical accomplishment. For instance, MIT has a rich history of students organising innocent pranks with a high degree of technical challenge, such pranks are called *hacks* (Peterson, 2011).

In our work, we interpret the word *hacking* as leveraging existing resources and modifying or appropriating them to serve an unintended purpose, often in an unexpected or clever fashion (see A. Williams et al. (2012) for a similar definition). Paradiso et al. (2008) and B. Hartmann et al. (2008) identify three underlying motivations for such a process:

- Hacking is used to overcome resource limitations and short deadlines: reappropriating an existing design is cheaper than creating a new design from the ground up.
- Hacking allows ideas to be fleshed out before they are advanced enough to merit formal development.
- Hacking serves as a source of inspiration, and by frequently engaging in hacking activities, designers build up a mental database of solutions.

Hardware hacking has become increasingly difficult due to continuing efforts of miniaturization, automation, and robotization in the manufacturing industry (Paradiso et al., 2008). This is evidenced by the decreasing size of electronic components: resistors of  $0.4\text{ mm} \times 0.2\text{ mm}$  are now routinely used in products. Current trends in consumer products include using glue instead of screws to keep housings closed and repairing broken products is being actively discouraged by some companies. Still, hardware hacking has seen a resurgence in recent years due to internet culture and the rise of open source hardware communities (Paradiso et al., 2008).

An interesting example is the phenomenon of IKEA hacking (Rosner, 2009). While hackers work alone, instructions for their hacked furniture is shared online. This is made possible by the fact that IKEA products are globally available, fulfilling a prerequisite of a reproducible design. In the article, one of interviewed IKEA hackers also noted the following: *“being inexpensive means IKEA products are not thought of as “precious,” so it’s psychologically and financially easier to tinker with them.”*



Human-Computer Interaction (HCI) research itself has a rich hacking history. As A. Williams et al. (2012) notes, most development and implementation activities in interaction research can be summarized as hacking. This is unsurprising, as development in academic research is typically limited to build a working proof of concept, and not to design a fully-finished product. Here, hacking is a valuable method considering the pressure to meet deadlines and the budget limits for prototyping.

As a final point, hacking as an activity in itself is also a point of study, both in theory and in practice. For instance, Von Hippel and Paradiso (2008) link hacking to lead user innovation from a theoretical perspective. By contrast, Zappi and A. McPherson (2014) apply hacking in the form of circuit bending as an integral design aspect of a novel musical instrument.

### 1.1.4 MAKER MOVEMENT

The maker movement is a name for the contemporary trend of DIY revival. The movement is closely associated with Make Magazine and their Maker Faires. Dale Dougherty, one of the key figures of Make Magazine, states that the magazine began “with the observation that people were hacking physical things again” (Dougherty, 2008). Section 1.1.3 explains the background of hacking in DIY. Previous sections show that this rise in “physical hacking” is made possible largely due to the emergence of open source hardware. Hacking commercial products has become much harder due to the ever-increasing complexity and miniaturization of consumer products.

Based on this insight, Dougherty, an executive at O'Reilly media (a publisher of technical manuals and programming books) decided to launch a new magazine in 2005. The magazine would draw inspiration from mid-20<sup>th</sup> century tinkering magazines of, such as *Popular Mechanics* (Dougherty, 2012). The word “maker” was chosen because the term is broad and very relatable. Not many people identify with words such as inventor or hacker. On the other hand, as Dougherty (2012) notes, everyone is a maker of something, be it cooking, gardening, or knitting. A year later, in 2006, Make Magazine organized the first Maker Faire in San Mateo, California. The Maker Faire was conceived as sort of show-and-tell event where makers could meet one-another to demonstrate and discuss their work (Dougherty, 2012). Since then, hundreds of Maker Faires and mini Maker Faires have been organized worldwide.

The maker movement shares many characteristics with older DIY trends. Most importantly, these trends emphasize building physical objects. However, the maker movement also has a number of characteristics that were not prominent in the preceding trends. Hatch (2013) argues that in addition to making, the movement also celebrates values such as sharing, collaboration, education, community and entrepreneurialism.

The internet has had a tremendous impact on the maker movement. It offers an accessible medium to share one's work with the world and it facilitates collaboration without physically meeting. Many online communities exist, each with their own specific focus.

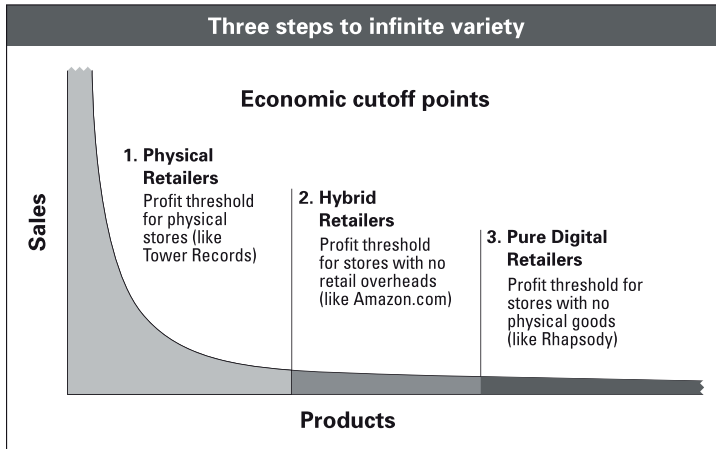
Examples include Instructables, an instruction sharing website; Etsy, a marketplace for homemade items; and Ravelry, a community for knitters and crocheters (Kuznetsov and Paulos, 2010). Online platforms greatly facilitates collaboration, especially when dealing with niche projects where one might not find collaborators closely.

It is notable that sharing and collaboration does not solely take place online. In addition to the aforementioned Maker Faires, recent years have also been marked by the rise of FabLabs, makerspaces, and hackerspaces, which serve as local community hubs for maker culture. FabLabs are publicly accessible spaces where tools such as laser cutters, 3D printers, and Computer Numeric Control (CNC) machines are made available. FabLabs usually charge a membership fee, similar to a gym membership, though many are open to the general public on select moments. The first FabLab was started more than ten years ago as an offshoot from MIT's Centre for Bits and Atoms as a way to empower users to "make almost anything" (N. A. Gershenfeld, 2005; Mostert-Van Der Sar et al., 2013). As of 2016, the Fab Foundation lists 715 FabLabs worldwide, spread over five continents<sup>5</sup>.

Many FabLabs serve a role in Science, Technology, Engineering and Math (STEM) education and outreach. Troxler (2013) shows that two out of three FabLabs are linked to educational institutions such as universities and colleges. Blikstein (2013a) draws parallels between the evolution of programming in education and that of design/engineering in education: "What Logo did for geometry and programming – bringing complex mathematics within the reach of schoolchildren – fabrication labs can do for design and engineering. Digital fabrication is Logo for atoms." Blikstein also points out that STEM skills are quickly becoming a new form of literacy due to the evolving needs of society (Blikstein, 2013a). Mostert-Van Der Sar et al. (2013) argue for the place of FabLabs in higher education, with design education in particular. The researchers show that FabLab environments enable rapid design iterations of physical artifacts, make a comparison to the *agile* method in software development, and argue that iterations are essential in user-centered design. Furthermore, FabLabs enable design students to go beyond purely mechanical prototypes, creating opportunities for incorporating microelectronics and programming into product prototypes. Martin (2015) shows that the maker movement is not only valuable for higher education, but also holds many opportunities within K-12 education: maker-centric activities mesh well with new curricular goals, and the act of making has an important transformative and empowering dimension. Finally, many students experience making as intrinsically pleasurable activity, and this intrinsic motivation for education is a factor that is lacking in traditional education (Honey and Kanter, 2013).

In *The Maker Manifesto*, Hatch (2013) recounts many stories of how TechShops – and the maker movement by extension – have dramatically lowered the barrier of creating and commercializing a physical product. Hatch summarizes that in the past, one would need an investor to develop a product due to the large amounts of capital required. With the rise of the maker movement, small products can be developed for under \$1000, *disposable* money. Hatch concludes that the maker movement has tremendous economical value because it allows disposable income, as opposed to investment capital, to be used to create new businesses.

<sup>5</sup><http://www.fablabs.io/labs>



**Fig. 1.3** Model of long tail economics. Adopted from Anderson (2008).

A similar sentiment is echoed by Anderson in his book *"Makers: the new industrial revolution"* (2012). Anderson frames entrepreneurialism within the maker movement as an example of long tail economics (fig. 1.3), a model which he previously introduced as an explanation for the success of internet companies such as the online book retailer Amazon and the music streaming service Spotify (Anderson, 2008).

The long tail model states that new business models are focusing on serving many different, small niche markets instead of serving few popular markets (Anderson, 2008). An Amazon employee summarizes the phenomenon as follows: "We sold more books today that didn't sell at all yesterday than we sold today of all the books that did sell yesterday." In other words, the majority of Amazon's revenue did not come from bestsellers, but rather from their enormous catalog of obscure books. The internet is a major factor here: whereas traditional book store have only so much shelf space available, the virtual shelf space of an online retailer is unlimited. In his book, Anderson (2008) gives many more examples of successful long tail companies. Music services such as iTunes and Spotify can offer a vast music catalog at negligible cost, whereas a traditional CD store can only offer the most popular music for sale. Google makes most of its advertising revenue from many small ad campaigns targeted at specific user groups. Ebay offers its users a marketplace for the most unusual objects, and most of the goods sold on Ebay cannot be found at big traditional retailers.

Anderson (2012) and Hatch (2013) point at the maker movement as one of the enablers of the long tail economics in hardware. When manufacturing one widget is as efficient as manufacturing a hundred, targeting new niche markets becomes feasible. This is the case with many digital manufacturing technologies. Because there is no tooling involved, these technologies are ideally suited for small-scale manufacturing. Anderson (2012) argues that digital manufacturing is a disruptive technology: whereas long tail economics of digital goods was enabled by the internet, the long tail of things will be enabled by cheap and

ubiquitous digital manufacturing technologies.

### 1.1.5 DIY IN RESEARCH

Do-it-yourself has a rich history of academic investigation. Particularly in the field of human-computer interaction, DIY is featured prominently. Perhaps, this may be explained by the field's ties with hacker culture. Academics have approached the topic both from a theoretical perspective as well as from a practical side. Previous sections have already touched upon the interplay of research and hacking, open source hardware, and the maker movement. This section will expand upon this topic with additional sources that were influential in our work.

Already in 2001, Von Hippel (Hippel, 2001; Hippel and Katz, 2002) hinted at the innovation potential of user toolkits in niche application areas and “markets of one”. Von Hippel describes the challenge of innovating in terms of “*sticky*” information; information that is difficult and costly to transfer from the user to the designer. For example, in the context of robot-assisted therapy, a therapist (the user) might be well aware of what robot aspects (embodiment and functionalities) are important to a specific patient. A roboticist however would not be aware of all these aspects. As Von Hippel explains, toolkits provide a way to bridge this asymmetry of information: it makes sense to give users the tools to build their own solutions instead of designing a solution for them.

Within the field of HCI, there already exists a large body of work on toolkits (Bdeir, 2011; Chung et al., 2013; Dietz et al., 2014; Oh and Gross, 2015; Simon et al., 2014). As shown by D. A. Mellis, Jacoby, et al. (2013), traditional toolkits do suffer from a number of drawbacks: users are limited to the set of modules that the designers of the toolkit have provided. Furthermore, the shape of the modules imposes a constraint on the shape and aesthetics of the artifacts designed with the toolkit. Instead, (D. A. Mellis, Jacoby, et al., 2013) propose “untoolkits” as a potential answer: toolkits that are interpreted more as a design method rather than a fixed set of building blocks. Examples of untoolkits include (Bouchard et al., 2015; Megaro et al., 2015; D. A. Mellis, Jacoby, et al., 2013; Thomaszewski et al., 2014). Yet another approach is to design an artifact with the specific intent of being modified by the end user. Examples of such hackable devices include (Forsslund et al., 2015; D. A. Mellis and Buechley, 2014; Zappi and A. McPherson, 2014).

Still, modularity in toolkits should be interpreted as a spectrum, with generic, universally-applicable modules at one extreme, and prescriptive, specialized modules at the other end. The generic approach allows a toolkit to be used in a wider range of situations, at the cost of being more difficult and time-consuming to use. Prescriptive modules, on the other hand, are easy to use, though more limited. It is impossible to identify a single, correct position in this spectrum, as the answer is ultimately highly dependent on context, user, and application. Consequently, the appropriate balance should be determined for each toolkit independently.

Tangible interaction research served as a valuable source of inspiration for our own work.

One prominent example is the work of J. S. Silver: the Drawdio and the Makey Makey (2014). Silver's work bears similarities to Mellis' concept of untoolkits, though Silver frames the idea as "improvised interfaces", as lenses which let you see the world as a toolkit. The first example – the Drawdio<sup>6</sup> – consists of a simple circuit mounted to a pencil that generates sound based on resistance. Because the graphite inside a pencil is electrically conductive, a user can create music by drawing. The second device – the Makey Makey<sup>7</sup> – lets users turn any conductive object, such as a banana or a blob of Play-Doh, into a keyboard button, allowing users to rapidly create new computer interfaces. With both of these examples, simple tools are used to design rich tangible interfaces without the need for programming. The system's limitations serve as a source of inspiration for users.

Finally, we remark that there is a deep and rich history of DIY within academia. This is not surprising, throughout history, scientists have been responsible for constructing their own tools to further their research. Think of how Galileo used self-built telescopes to discover the moons of Jupiter, or how Van Leeuwenhoek's microscope led to the discovery of bacteria. The tradition continues to this day, and as A. Williams et al. (2012) note, researchers are essentially hackers at heart. Even now, the aforementioned trends of DIY, hacking, and the maker movement are embraced by academics as a mechanism for the democratization of research and invention. Taking this one step further has led to the citizen science movement (Paulos et al., 2009; Phillips et al., 2014), as well as the push for open source lab equipment (Herrmann et al., 2014; Pearce, 2012; Pearce, 2013).

## I.2 LEARNING & CREATIVITY

This section looks at learning in its broader context. The term "learning" is often associated with schools and other institutions for education, however it is a part of our daily lives. Modern society requires us to continually learn. Learning can be formal or informal, student/teacher or self-paced. Even now, as technology permeates every aspect of our daily lives, life-long learning becomes an ever-important skill. This section details the influences of learning and creativity that were important over the course of this project.

### I.2.1 STEM EDUCATION

In recent years, western policy has been characterized by an emphasis on transitioning toward knowledge-based economies. In 2000, the European Union signed a policy plan called the Lisbon Agenda, aiming to make the EU "the most competitive and dynamic knowledge-based economy in the world capable of sustainable economic growth with more and better jobs and greater social cohesion" (EU, 2000). This was followed by the Europe 2020 strategy in 2010, which lists investing 3% of the gross domestic product in research and development as one of the main policy targets (EU, 2010).

<sup>6</sup><http://www.drawdio.com>

<sup>7</sup><http://www.makeymakey.com>

However, as western societies orient themselves, the problem of insufficient science and technology professionals is exacerbated. For instance, in Flanders, there is an enormous imbalance in the job market for engineers, with demand exceeding the number of engineers by a ratio of approximately four to one (VDAB, 2012). The problem is not limited to engineering degrees, but extends to a wide spectrum of scientific, technical, and technological jobs. The term STEM – an acronym standing for Science, Technology, Engineering, Math – is often used to summarize these knowledge domains.

In the past ten years, there has been an explicit focus on STEM education in western countries such as Belgium. However, this policy focus is not just driven by the need for people with STEM degrees in the job market. The intended reach for this policy change is much broader. The focus on STEM skills is driven by a society in which technology takes an increasingly significant role in everyday life (Schmidt and Cohen, 2013). This does not only result in a high demand for STEM skills on the job market, it also means that technological literacy is increasingly becoming a necessary life skill for everyone. This distinction is also reflected in a policy statement from the Royal Flemish Academy of Belgium for Science and Arts (Veretennicoff et al., 2015).

Stimulating STEM literacy and interest in students from developed countries is not as straightforward as it seems. The ROSE project (Schreiner and Sjøberg, 2010) revealed a number of interesting dimensions toward the problem of science and technology education. To begin, students from both developed and developing countries typically agree that science and technology make life better. However, the study revealed an interesting paradox: the more developed a nation is, the less students are interested in science and technology class topics and the less interested they are in pursuing STEM careers. The study also revealed remarkable gender differences. Overall, girls are much less inclined to show interest in science and technology. Interest in different topics within STEM also show a strong influence of gender: girls are more interested in topics such as health and environment, whereas boys are more inclined toward topics such as mechanics and electricity.

A distinction should be made in current efforts in technology education: there is a difference between technological literacy (important for everyone) and technological competence (important to do your job) (Blikstein, 2013a). The first goal, technological literacy, can be summarized as the basic knowledge of the technological principles that govern our daily lives. As technology is becoming increasingly ubiquitous and permeates all aspects of our lives, it is important for everyone to have basic technical skills (Pearson and Young, 2002). The second goal, technological competence, refers to the skills of professionals to manipulate technology to suit a specific purpose. Technological competence refers to the skills that are required to do one's job. Not all jobs require technological competence, though the job market has a large demand for such profiles. One challenge remains prevalent in engineering and technology education: knowledge evolves so quickly that the specific domain knowledge one learns in school can become outdated by the time one graduates. Hence the importance of learning how to learn, and not just memorizing specific facts.

More recently, educators have begun to argue that the acronym STEM should be replaced

by STEAM. The *A*, referring to *arts*, emphasizes the importance of creativity. Maeda (2013) argues that true innovation only happens when convergent thinking is combined with divergent thinking, and that science and arts work better together than they do apart. Piro (2010) argues that art represents a sizable portion of the United States economy, and that artistic involvement directly leads to better proficiency in creativity, collaboration, communication, and critical thinking – all touted as hallmark 21st-century skills. Equally important is that arts instruction emphasizes that problems can have many solutions, a nuance that is often overlooked in conventional education. J. W. Bequette and M. B. Bequette (2012) argue that art education should be broader than art *eo ipso* and that functional art, such as product design and graphic design, could be incorporated in the curriculum of STEAM education. They also argue for the similarities between arts and engineering education: both are born from ill-defined, real-world problems, and both topics lend themselves well to problem-based learning approaches. As evidenced by the preceding paragraphs, STEAM covers a wide breadth of skills, topics, and disciplines. In the rest of this work, we focus in particular on incorporating elements of technology (*T*), engineering (*E*), and art (*A*) education into our work.

### 1.2.1.1 METHODS

Studies show that certain teaching strategies can be used to foster STEM participation and achievement. Traditionally, STEM topics are taught in classrooms using an *ex-cathedra*, instructionist approach, where the teacher introduces new knowledge and the students listen and follow. However, Prince (2004) shows that switching from a traditional education approach to *any* form of active learning leads to a better learning outcome in engineering education. Similar effects are demonstrated by Hake (1998), who analyzed the effects of interactive engagement in physics education leads to an increase of 108% in learning outcomes.

One of the most prominent pedagogic methodologies within STEM education is *constructionism*, as introduced by Seymour Papert in his seminal work *Mindstorms: Children, Computers, and Powerful Ideas* (Papert, 1980). As mentioned earlier, one of the predominant modes of teaching used in schools are *instructionist* methods. In this format of teaching, there is a clear distinction between the teacher, serving as the provider of knowledge, and students as the recipients. Papert (1980) introduces constructionism as an alternative to this approach.

Constructionism is based on the constructivist theory that “learners construct mental models to understand the world around them”. Constructionism is a student-centric approach where students use knowledge they already possess to acquire new knowledge. The approach emphasizes that learning happens most effectively when it is connected to making tangible objects, though the approach is described as much deeper than a simple *learn-by-making* formula (Papert and Harel, 1991). In essence, instructionism can be summarized as the “transfer of knowledge to students”, whereas constructivism and constructionism entail the “production of knowledge by students”. Though remarking upon the irony of giving a definition of constructionism (Papert and Harel, 1991), Papert offers the follow-

ing description (Papert, 1986):

*“The word constructionism is a mnemonic for two aspects of the theory of science education underlying this project. From constructivist theories of psychology we take a view of learning as a reconstruction rather than as a transmission of knowledge. Then we extend the idea of manipulative materials to the idea that learning is most effective when part of an activity the learner experiences as constructing a meaningful product.”*

The work of Papert is centered around using computers as a medium for education, though computers are not a prerequisite to constructionism and they are also not the only viable medium: “computers figure so prominently only because they provide an especially wide range of excellent contexts for constructionist learning” (Papert and Harel, 1991). Within the framework of Papert’s research, Logo (a programming language designed for children) and Turtle graphics (a tool for using programming to make images) were developed. Papert likened the process of learning Logo to living in “mathland”; it is as natural to learn math this way as it is natural to learn French while living in France.

The well-known LEGO Mindstorms robotics kits are also named in reference of Paper’s book, though the project was originally called LEGO/Logo. Recent years have been marked by the emergence of new media that is especially appropriate for constructionist learning. The work of Blickstein identifies digital fabrication (Blickstein, 2013a) and robotics and physical computing (Blickstein, 2013b) as especially promising media for applying constructionist approaches in STEAM education.

Another novel teaching paradigm for STEM education is called Project-Based Learning (PBL). In this student-centric pedagogy, students learn about a topic by exploring a certain open-ended problem. Fortus and Krajcik (2005) argue that traditional science education is built upon well-defined problems, such as predicting the ideal trajectory of a projectile. On the other hand, science in practice is predominantly occupied with ill-defined problems where “decisions are not clear-cut, where requirements can conflict, where optimization rather than ‘proof’ is needed” (Fortus and Krajcik, 2005).

In STEM fields, in particular, this method is gaining traction: examples include (Fortus and Krajcik, 2005; Kolodner and Camp, 2003; S. McPherson, 2014; Rockland et al., 2010; Törnkvist, 1998). As technology is evolving at an ever-increasing pace, it is no longer sufficient to just be able to memorize and reproduce factual knowledge. Instead, a deeper understanding of knowledge and the ability to apply knowledge and skills in a real world context are becoming increasingly important. It is these aspects, in particular, that the PBL approach excels at (Capraro et al., 2013).

Building robots is a popular project choice for the implementation of PBL in classrooms. The reason why it is such a popular choice can be explained by the interdisciplinary nature of the topic: robotics requires many different scientific, technical and technological skills, such as physics, electronics, mathematics and programming. It is an ideal subject because so many different courses can be linked to it (Johnson, 2003). Additionally, robots



themselves capture the imagination of children and teenagers, providing inspiration and motivation (Johnson, 2003).

The PBL approach in general and the use of robotics in education in particular have a number of other differences with more traditional ways of teaching. Whereas math problems typically have one, and only one correct answer, PBL emphasizes that most real world problems have many different solutions. With PBL, students learn to deal with these real world problems using creative problem solving, an important real-life skill. In addition to technical skills, PBL also allows the students to learn important social skills, such as communication, leadership, planning and cooperation (Bell, 2010).

### 1.2.1.2 CHALLENGES

Though these novel teaching approaches show great promise, they are not without their challenges. Historically, engineering education has shifted from professional engineer to the scientific engineer over the years (Tryggvason and Apelian, 2006). Theoretical classes have won out over the more expensive engineering labs and design classes (Feisel and Rosa, 2005). However, two influences have begun to reverse this trend: (1) as it turns out, graduates are not prepared for real engineering work, (2) prototyping equipment has become dramatically cheaper, creating prototypes can now happen in days instead of months (Blikstein, 2013a).

Even though active teaching methods are slowly starting to gain a foothold in STEM education, the underlying issues that have lead to *the scientific engineer* remain a challenge. Active teaching approaches are more costly than instructionist approaches: they require more teacher time and training, they require smaller class groups, and they require more equipment and infrastructure.

Constructivist and constructionist teaching approaches also have far-reaching effects on classroom dynamics. When switching from well-defined classroom problems to ill-defined, open-ended problems, the teacher needs to assume a new role: the role of a mentor or coach. Many teachers are unaccustomed to this new role. The open-ended nature of such active teaching approaches mean that there is a very real chance that students will have questions that a teacher cannot answer. Again, this is not the case with traditional teaching, where the teacher can reasonably expect to be able to answer all questions from students.

While using robotics for PBL offers a promising alternative to the traditional teaching methods, implementing this on a large scale in education poses several challenges. Mataric et al. (2007) sums up five big challenges:

1. Lack of teacher time.
2. Lack of teacher training.
3. Lack of age-suitable academic materials.

4. Lack of ready-for-use lesson materials.
5. Lack of a range of affordable robotics platforms.

Especially the cost remains a barrier in the implementation of robotics in education (Gonzalez-Gomez et al., 2012; Johnson, 2003; Mataric et al., 2007; Mondada et al., 2009; Riojas et al., 2012). Schelly et al. (2015) argue that as a general rule, STEM education is more costly than traditional education, and the trend of decreasing budgets for education combined with the rising costs of lab equipment leads to a precarious situation for STEM education.

Besides these challenges, gender issues remain relevant in the context of technology education. Presently, popular STEM topics such as robotics and other technological hobbies are usually associated with boys, as discussed in the ROSE project (Schreiner and Sjøberg, 2010). Girls are often subtly discouraged and told to pursue other interests (Modi et al., 2012). As a result, women are underrepresented in STEM-related fields. Studies have shown that while girls will not always focus on the same aspects of building robots as boys, they show just as much interest in the topic (Johnson, 2003). Consequently, robotics – if properly approached – can serve as a way to increase the number of women in technical and technological fields (S. Hartmann et al., 2007).

## 1.2.2 ROBOT KITS AS LEARNING MATERIAL

As touched upon in the previous section, the topic of robotics is frequently chosen by teachers as the subject of STEM-focused PBL. The reason for this is obvious: as Benitti (2012) and Johnson (2003) show, teaching robotics is a very effective way of motivating and integrates many different knowledge domains of the curriculum. Furthermore, it also stimulates students' social and teamwork skills. As a secondary aspect, robots are something that captures the imagination of many children, which serves as a motivator (Johnson, 2003). Examples of robots being used in STEM education are plentiful, and cover both K-12 and university-level instruction (Benitti, 2012; Carbajal et al., 2011; wyffels, Hermans, et al., 2010).

Broadly speaking, there are two ways to implement robotics in an educational context: (1) starting from an existing robotics kit or (2) by building robots from scratch. Building a robot from scratch tends to be much more difficult. Consequently, kits tend to be more popular in classrooms, especially when younger students are involved. Robot kits provide everything needed to make a functional robot, such as building elements, motors, sensors, instructions, a programmable control unit, and the software to program the robot. While this is a great way to get up and running quickly, the approach does include several disadvantages:

- Kit components tend to be more prescriptive and less flexible.

- It can be hard to interface proprietary kits with components such as standardized parts, components made by third parties, or off-the-shelf sensors and actuators.
- Finding or buying replacement parts is not always possible.
- Not all components are used or needed in a robot, so you are paying for components you do not need.

That being said, these systems are a great starting point as they provide a set of compatible building blocks and electronics, software for easy (graphical) programming, instructions and a community network.

LEGO Mindstorms (fig. 1.4) is a popular example of a commercial robot kit. Mindstorms is built around a programmable microcomputer brick that can control up to 3 motors and read up to 4 sensors. LEGO Technic style bricks are used to build the structural parts of the robot. Because of this, building blocks from other Lego sets can be easily incorporated, expanding the potential level of intricacy of the robots. Mindstorms is already being used in many classrooms, both in K-12 (W. Church et al., 2010; K. Williams, Igel, et al., 2012) and at university level (Brandt and Colton, 2008; Ranganathan et al., 2008). The popularity of Mindstorms translates to a plethora of resources available for educators, including books, robot contests, online communities and workshops that are built around the Mindstorms ecosystem. Despite these advantages, one of the main downside of commercial robot kits lies in their closed, proprietary nature.



**Fig. 1.4** LEGO Mindstorms EV3 kit



**Fig. 1.5** Thymio. Adopted from Riedo, Chevalier, et al. (2013).

While commercial kits dramatically accelerate the design process, they also limit the maximum potential of the robots. The number of motors in LEGO Mindstorms is limited because the programmable brick can only drive three motors. While this is enough for simple differential-drive robots, more complex robots cannot be created. Sometimes, students want to use more motors, but run into this system limitation. Users are also limited to the components offered by the kit, and interfacing with third-party components is hard,

actively discouraged, or outright impossible. Finally, at a price of around 400 EUR for the base set, the platform can be quite costly for certain users. Providing enough robot sets for an entire class can quickly become an expensive affair. Larger schools can alleviate this by buying enough sets for one group and then pass them around between the different class groups, though this is not always feasible, especially for small schools or organizations.

A second category of off-the-shelf educational robots are self-contained driving robots, such as the Thymio (Riedo, Rétornaz, et al. 2012, fig. 1.5) and the Edison<sup>8</sup>. This type of educational robot can be used as-is, without any construction or assembly by the user. However, some robots – including the Thymio and the Edison – are designed to be extended using craft materials or LEGO bricks. These robots are affordable and robust, two important aspects in an educational context. However due to their prescriptive design, they tend to emphasize programming activities over construction and physical experimentation.

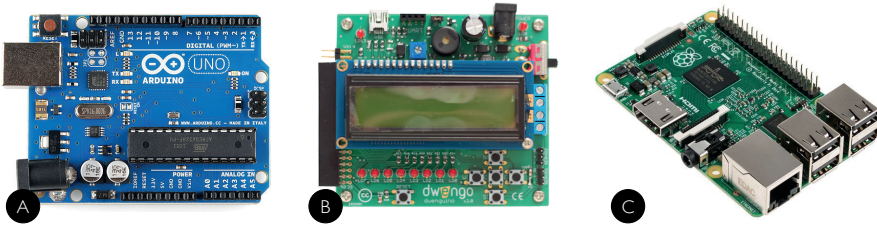
DIY and OSHW variants of this type of robot also exist. For instance, the MiniSky-Bot (Gonzalez-Gomez et al., 2012) is a small differential-drive mobile robot that can be built using 3D-printed components and off-the-shelf parts. This approach is interesting because it offers beginners a working design to start with. The MiniSkyBot can be used for similar educational programming activities, such as programming a line-following behavior. However, because the robot is open source, it can be used as a starting point for hacking and extending functionalities. Doing so brings students into contact with real engineering principles and challenges, and represents a more holistic approach to STEM education.

In recent years, making a robot from scratch has become much easier. As described in section 1.1, trends such as the open source hardware movement and the maker movement have had a tremendous influence in democratizing many technologies, including robotics. Recently, many different products and platforms have emerged that implement elements of a robotic system. These elements can be categorized into three groups: software, electronics and hardware components. Historically, software has been the easiest to share as Open Source because collaboration can be done easily over the internet, because development tools are readily available and because there is virtually no cost associated with replication or adaptation. Online platforms such as GitHub greatly facilitate this process (Dabbish et al., 2012). However, in recent years the electronics and hardware projects have flourished under the influence of DIY, open source, and maker trends.

One of the core requirements for a functional robot is a programmable controller that can read sensors, process information and drive actuators. Many options are available for robots built from scratch, making it possible to tune the electronics to the specific functionality needs of a robot. Boards from the Arduino ecosystem (discussed in section 1.1.2) are a popular option for small scratch-built robots. Examples of Arduino-based robots include Araújo et al. (2014), Gonzalez-Gomez et al. (2012), and Juang and Lurrr (2013). The combination of a low cost board and the ease of use of the software have

<sup>8</sup>Meet Edison - A Cheap Programmable Educational Lego Robot Kit – <http://meettedison.com>

made Arduino a very popular platform, especially among hobbyists.



**Fig. 1.6** Examples of programmable controllers: (A) Arduino Uno, (B) Dwengo, (C) Raspberry Pi.

The Arduino boards are not specifically designed for use in robotics, but their functionality can be extended through daughter boards called shields. Shields are attached on top of an Arduino board and provide the board with extra functionality, such as circuitry to control DC motors. Many of the Arduino and Arduino compatible boards use the same physical pin layout; because of this the shield form factor has become a de facto standard. Many different shields exist, providing a plethora of possible functionality, made by either the Arduino organization or, more often, by third party vendors. The size of the Arduino ecosystem gives robot designers a large degree of flexibility. On the other hand, educators face the problem that choice can be overwhelming; the starting point for building robots with Arduino is not always clear. This corresponds with problems identified by Mataric et al. (2007), such as the “lack of teacher time” and the “lack of teacher training”.

A more specialized example of a controller board suitable for educational robotics is the Dwengo. The project aims to provide an easy to use platform for getting started with microcontrollers, electronics and programming (wyffels, Hermans, et al., 2010). Originally started in 2009 at Ghent University as a microcontroller experimentation board for internal use, it was developed further when its potential benefits for education became clear. Unlike Arduino boards, which typically aim to provide a very low cost bare-bones board, the Dwengo board includes features to facilitate building robots. The board includes a display, input buttons, 2 servo connectors and a 2 channel DC motor driver. In addition to the electronics board, the Dwengo project also offers a set of step-by-step tutorials created specifically for secondary school education. Like with the Arduino, building robots with Dwengo is more complicated than using an off-the-shelf kit. On the other hand, this approach is more flexible and brings students into contact with real-world engineering principles.

Finally, the Raspberry Pi<sup>9</sup> is a low cost single board computer, developed to promote the teaching of programming and computer science in education (Mitchell, 2012). This board is different from previous examples in that it runs a full Linux OS. In practice, the board can do more advanced things that an Arduino can’t, such as running computer

<sup>9</sup>Raspberry Pi – <http://www.raspberrypi.org>

vision algorithms and communicating with the internet. However, the Raspberry Pi is not suitable for real-time tasks due to the processing overhead of the OS. Boards such as the Arduino and the Dwengo are better suited for such tasks. In addition to USB and Ethernet ports, the board also has a General Purpose Input/Output (GPIO) pin header that gives access to low-level peripherals. The board's low cost combined with the access to low-level peripherals have made the board popular with hobbyists.

The Raspberry Pi lacks many of the features that are required to build a robot, such as the ability to control DC motors. Luckily, this functionality can be added using daughter boards, much like Arduino shields. While more complex to use than a simple microcontroller, the advanced capabilities of the board allow for experimentation with technologies such as AI and computer vision.

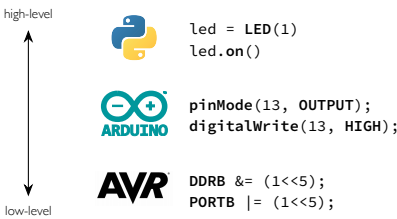


Fig. 1.7 Text-based programming languages.

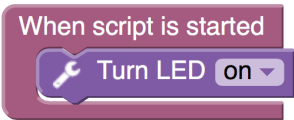


Fig. 1.8 Visual programming language made with Blockly.

The second aspect of a robot is its software, which the behavior of the controller. Many different programming languages exist. However, most microcontrollers can only be programmed in one or two languages. Traditionally, a robot's microcontroller is programmed using a low-level, text-based programming language such as C. While powerful and resource-efficient, C is a complex language and is not beginner-friendly. The Arduino boards are programmed in C, though the programming process is made much easier through carefully designed libraries. The design of the Arduino Application Programming Interface (API) is based on Wiring (Barragán, 2004), which in turn is based on Processing (Reas and Fry, 2007), a programming for creative coding. The Arduino libraries allow cryptic port manipulation commands to be replaced by more intuitive code, as illustrated in figure 1.7.

The Python programming language is also frequently used in education (Stajano, 2000). This language can also be used to program educational robots (Blank et al., 2003). Unlike C, Python is a high-level programming language, meaning that it has a degree of abstraction over the technical details of the computer. As a result, the language is much easier to use. The language is designed to run on full scale computers, but also runs natively on single-board computers such as the Raspberry Pi. Recently, efforts have been made to port the programming language to low-cost microcontrollers. MicroPython<sup>10</sup> implements a subset of the Python programming language and runs directly on microcontroller boards such as the ESP8266 and the BBC Micro:bit.

<sup>10</sup>MicroPython – <http://micropython.org>

The second group of programming languages used in education are visual programming languages. These languages let users create programs through a graphical drag-and-drop editor. Text-based languages can be quite daunting for novices. Not only is it hard to translate human language concepts to algorithms that a computer can understand, one also has to take care that the syntax of the program is correct. Graphical programming languages alleviate the second problem, while also providing an interface that is more appealing to children than a text editor. On the other hand, programming in a graphical languages is slower and more tedious than using a text-based language. Additionally, graphical programming languages are not suitable for large scripts, as they take up a lot of screen space. Visual programming languages should be seen as a stepping stone, making real-world programming accessible to a large audience of users. Professional programmers will most likely continue to use text-based programming languages because of their efficiency, even though they are more difficult to learn. Some graphical programming languages offer the option to generate text-based code output. This can be helpful to transition from visual to text-based programming.

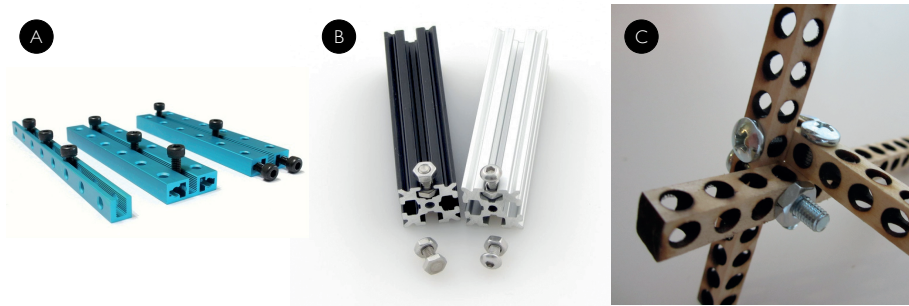
Scratch is a notable example of a graphical programming language that is designed for education. Developed at MIT Media Lab in 2006, the language aims to teach children the principles programming through the creation of simple games and interactive movies (Resnick et al., 2009). The language was very successful; as of this writing, more than 17 million Scratch scripts have been shared by users<sup>11</sup>. Scratch features a display area, onto which different sprites can be placed, and a programming area, onto which puzzle-like programming blocks can be placed. These puzzle blocks can represent simple commands ("move 10 steps"), or more complicated control structures, such as loop statements ("repeat ...until") or conditional statements ("if ...then ..."). The blocks can be snapped together to create a logical sequence of actions, similar to the sequence of statements one would find in traditional code. Because of the different shapes of the different types of puzzle-like blocks, they can only be combined in a way that makes sense, eliminating the possibility of syntax errors.

Scratch is very much oriented at computer-centric use. While some options exist to make Scratch interact with the outside world, these options are limited and Scratch is better suited for computer use only. However, the project has inspired a number of derivative languages that are suitable for creating programs that interact with the physical world. Google's Blockly is a library that greatly facilitates the development of Scratch-like programming languages. Using the Blockly Application Programming Interface (API), one can easily define its own set of blocks in order to create a fully customized graphical programming language. Blockly cannot be used to program robots directly, but it does provide a very convenient way to design a language that can be used for that purpose. Examples include Goud et al. (2015), Pacheco et al. (2015), and Saleiro et al. (2013), as well as the visual programming interface in this work. Further examples of derivative languages include ArduBlock<sup>12</sup> and ModKit (Millner and Baafi, 2011), two graphical languages to program microcontroller boards such as the Arduino.

<sup>11</sup>Scratch – <http://scratch.mit.edu>

<sup>12</sup>ArduBlock – <http://blog.ardublock.com>

The third and final element is the embodiment of a robot. Hobbyist robot makers tend to rely on a wide variety of techniques when it comes to building the physical embodiment of their robots. Some repurpose old toy components, some make their robot out of cardboard and duct tape, some even build their own metal chassis using advanced CNC machines. The democratization of digital manufacturing infrastructure in recent years means that many custom robots are now created using 3D printers and laser cutters. Still, there are projects that aim to facilitate the embodiment design process by providing a standardized construction system that is suitable for robotics. Some examples are shown in figure 1.9.



**Fig. 1.9** Examples of construction systems: (A) MakeBlock, (B) OpenBeam, (C) BitBeam.

MakeBlock<sup>13</sup> is a commercial building system aimed at building robots and other small-scale mechatronic toys. The system is built around aluminium beams. These beams can be connected using standard machine screws. The aluminium beams have a threaded slot, threaded ends, and two rows of equidistant holes. In addition to these beams, the system includes accessories such as motor mounts, servo mounts, angle brackets, and joining plates. The MakeBlock system is a great way to build rigid constructions for robots. The threaded slots make it easy to incorporate third-party components. The system uses the same hole spacing as LEGO Technic bricks, further improving compatibility. However, the system is relatively costly.

Another option is the use of miniature T-slot extrusions. T-slotted aluminium framing is commonly used in industry to build custom enclosures, test setups, or custom machines. The T-slot system is flexible and high-quality, though standard profiles are too large for small-scale educational robots, with 20 mm × 20 mm being the smallest size available from industrial automation component vendors. However, several companies have designed miniaturized versions of this industrial framing system, specifically aimed at makers and hobbyists. The first example is MakerBeam<sup>14</sup>, a 10 mm × 10 mm version of the T-slot system. OpenBeam<sup>15</sup> is another example, based around 15 mm × 15 mm cross-section. These systems were not specifically designed with robotics in mind, but

<sup>13</sup>MakeBlock – <http://www.makeblock.com>

<sup>14</sup>MakerBeam – <https://www.makerbeam.com>

<sup>15</sup>OpenBeam – <http://www.openbeamusa.com>



because other components can be connected so easily, they can be adapted to suit this task. One downside is the systems focus on static connections, but don't offer specialized components for constructing moving mechanisms.

A third and final construction system is GridBeam<sup>16</sup> and its variants. These systems use beams with equidistant holes as the main construction element. GridBeam uses 1.5 in square beams, a size that is suitable for constructing large objects such as furniture. The BitBeam<sup>17</sup> variant of the GridBeam system uses 8 mm square beams with holes spaced at 8 mm intervals, making it much more suited for building small scale robots. The 8 mm spacing matches that of LEGO Technic bricks, making integration with LEGO bricks trivial. OpenStructures<sup>18</sup> (Abel et al., 2011, p. 229) also falls within this category. The OpenStructures system is distinct in that it is designed to be scaled, making it appropriate for anything from small tools and devices, to furniture, to houses. All these systems are comparatively simple and can easily be replicated. Gridbeam-style components can be created through many different methods. Beams can be made using hand tools or with CNC machines, and the design (i.e. the dimensions) can be modified to suit the specific needs of each project.

Educational robotics have the potential to transform education from student observation and listening to active engagement through interactive hands-on lessons, guided by instructors and augmented by real-world examples and technology. However, the feasibility of a robotics-enhanced problem-based curriculum depends on the access to versatile robotics tools and well-organized tutorials. Generally speaking, there are two ways to build a robot: either by using a complete robotics platform, or by constructing a robot from scratch. Complete systems are easier to use, allow for quicker results and are better suited for young students. The downside is that they are generally more expensive and less flexible. Building a robot from scratch, in contrast, is much harder and is more suited for older students, but gives much better insight in the technology, is more flexible and can be much cheaper. In recent years, building a robot from scratch has become much easier due to numerous projects and products that implement certain aspects of a robot, such as hardware, software or electronics. These products and projects are linked to the recent advancements in DIY and maker movement trends. We believe that the DIY and Maker subculture can have a valuable impact on education, as it not only encourages young people's interest in STEM-related fields, it also fosters creativity and technological fluency.

In our experience, educators frequently choose an all-in-one robotics platform because they do not know of any alternatives or because they cannot find a clear starting point for alternative platforms. This choice is often further motivated because their regional colleagues tend to use the same platform, so there is a certain form of a support network. In our experience, classes that use a complete robotics platform, such as Mindstorms, tend to outnumber classes that build their own robots from scratch by a large margin. And while robots built with an all-in-one kit may perform better, students that build their own robot either completely from scratch, or by combining elements from the different systems as

<sup>16</sup>GridBeam – <http://www.gridbeam.com>

<sup>17</sup>BitBeam – <http://bitbeam.org>

<sup>18</sup>OpenStructures – <http://www.openstructures.net>

described above, tend to gain a much deeper understanding of technology, engineering and creative problem solving.

### 1.2.3 CREATIVITY

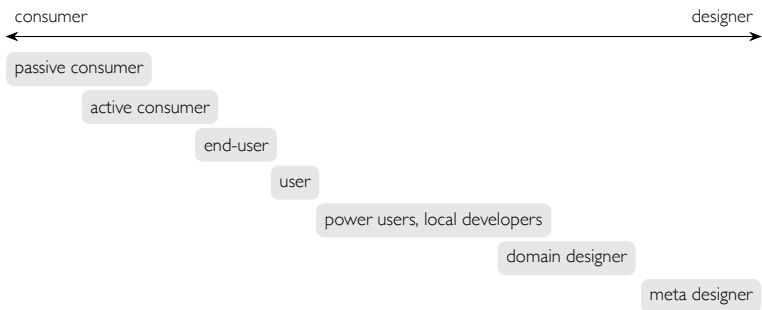
Recent changes in education and in society have placed an emphasis on the importance of creativity. Simply memorizing and reproducing knowledge is no longer sufficient for a productive and satisfying life. Simply put, advances in computing and information technology have greatly diminished the importance of memorization. Computers are much better than humans at storing and recalling data, they are also much better at calculation, however they are not (yet) capable of creative thought.

This focus on creativity is prominently present in online culture. Already in 2005, Lenhart and Madden reported that more than half of american teenagers post their own content online. Furthermore, in recent years websites centered around user-generated content have seen a rise to prominence. This includes platforms such as Etsy and Instructables (Kuznetsov and Paulos, 2010), as well as the emergence and popularity of social media.

Sanders (2006) argues that all people are creative to a certain extent, and that people have different creativity levels in different domains: “People have different needs for creativity in different domains of their life”. Furthermore, she discerns four basic levels of creativity, illustrating the difference between these levels using simple cooking analogies:

1. doing – *buying a prepackaged meal*
2. adapting – *adding an extra ingredient to cake mix*
3. making – *creating a meal using a recipe*
4. creating – *improvising a meal using ingredients from the fridge*

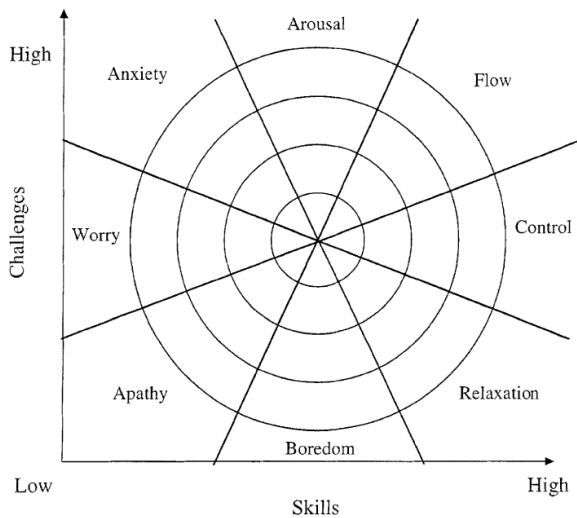
Fischer (1998) argues that the consumer/designer dichotomy is incorrect, and that reality is much better modeled as a spectrum (fig. 1.10). Furthermore, he argues that systems should be able to grow with the user. In other words, as the user’s knowledge of the system grows, the system should offer new opportunities for the user. The desire to create is linked with personal meaning. This leads to two problematic situations: (1) a person wants to be a designer but is forced to be a consumer, (2) a person wants to be a consumer but is forced to be a designer. Though Fischer’s work originates in the field of human-computer interaction, he argues that this idea “applies to cultures, mindsets, media, technologies, and educational systems in general” (Fischer, 1998). The question then becomes: if in the future, everyone will design their own things, what role will designers take? Within the field of co-design, some argue that designers will gradually evolve into a role of mediators stimulating creativity in others (Sanders and Stappers, 2008). Even though everyone has the potential to be creative, the act of being creative is more difficult than passively consuming. However, most people can be motivated by making the creative process easier. By matching the challenge to their skills, a mental state of flow can be reached.



**Fig. 1.10** The consumer/designer spectrum. Adapted from Fischer (1998).

**1.2.4 FLOW**

In positive psychology, the theory of flow (Nakamura and Csikszentmihalyi, 2002) offers an explanation as to why humans find certain activities intrinsically rewarding or satisfying. The model compares the skills of a person to the challenges posed by an activity. Different combinations of these two parameters result in different mental states, as shown in figure 1.11. For instance, an activity that is too challenging for the skills of a person may lead to a state of anxiety in that person. The theory of flow states that when the challenges are closely matched to a person’s skills, that they enter a special mental state called *flow*. This state is sometimes colloquially referred to as being ‘*in the zone*’.



**Fig. 1.11** Csikszentmihalyi’s model of flow. Adopted from Nakamura and Csikszentmihalyi (2002).

A person can enter flow if two conditions are met:

1. The perceived challenges match a person's capabilities
2. There are clear, proximal goals and there is immediate feedback on the progress toward those goals.

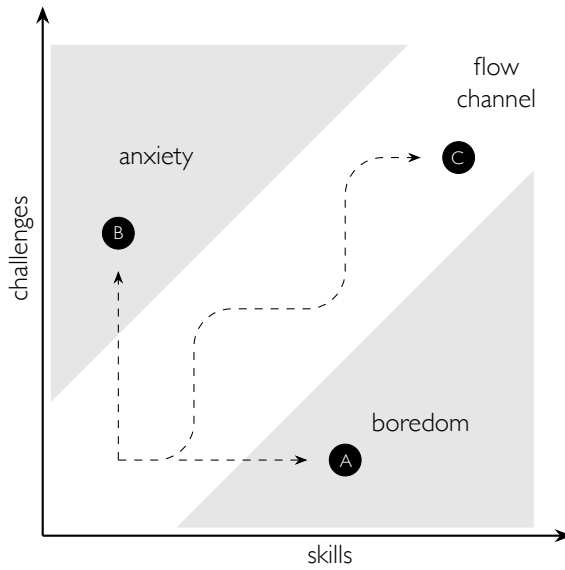
The state of flow is characterized by the following properties:

- There is an intense focus and concentration.
- Awareness is fully focused and devoted to the activity.
- One loses track of oneself as a social actor.
- There is a feeling of being in control.
- Sense of time is lost.
- The activity is perceived by the person as intrinsically rewarding.

The phenomenon of flow has been observed in both play and work contexts. It has also been observed across a wide range of activities, including art, science, sport, and literary writing. The experience of the state of flow is reported to be the same irrespective of boundaries such as age, gender, culture, or activity type. However, the state is based on an inherently fragile balance, as illustrated by figure 1.11. If a person's skill exceeds the challenge, they first relax, and eventually become bored. If, on the other hand, the challenge exceeds their skill, they first become vigilant and then anxious.

In their work, Nakamura and Csikszentmihalyi (2002) compare the model of flow to learning processes in pedagogy. The concept of matching challenges to skills bears remarkable similarities to the Zone of Proximal Development (ZPD) (Chaiklin, 2003; Vygotsky, 1978). This pedagogic model defines three concentric zones of development. The innermost zone comprises tasks a student can do completely unaided. The outermost zone comprises tasks a student cannot do at all. In between these two zones is the ZPD, which comprises tasks a student can do under the guidance of a more experienced teacher or peer. Vygotsky argues that learning occurs in this zone, and that given enough practice under guidance, a student will be able to complete the challenge unassisted. Similar to the state of flow, tasks located within the ZPD are difficult enough to be challenging, yet simple enough to be achievable.

From a designer's perspective, flow is an interesting tool to work with. Figure 1.12 shows a modified version of Csikszentmihalyi's model, depicted as a design space. To design an engaging product, the designer should pay attention to the narrow zone where flow is possible. The longer a user is engaged with a product, the more proficient they will become at using that product; a natural consequence of learning and experience. If then, the challenges remain the same, the user will soon find themselves in a state of boredom (fig. 1.12 A). On the other hand, if the full gamut of challenges are presented immediately, it can result in a state of anxiety (fig. 1.12 B). Consequently, in order to avoid boredom



**Fig. 1.12** A designer's perspective of flow. Adapted from Nakamura and Csikszentmihalyi (2002).

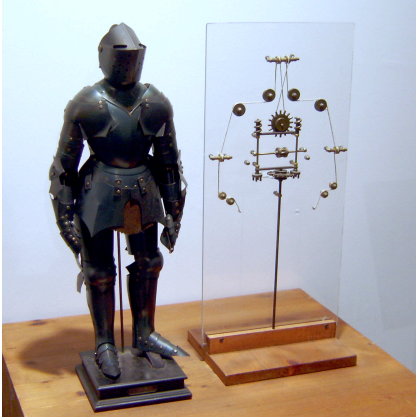
and anxiety, the system should gradually present new challenges as the user becomes more proficient (fig. 1.12 C).

Note that there are parallels between the theory of flow and the work of Fischer (1998) and Sanders (2006), despite originating from different fields of study. Flow theory states that an activity should become increasingly challenging as the user becomes more skilled at said activity. Fischer (1998) argues that systems should be able to grow with the user, fulfilling one of the requirements of flow. The same sentiment is echoed by Raymond (1999) when discussing open source communities: “Human beings generally take pleasure in a task when it falls in a sort of optimal-challenge zone; not so easy as to be boring, not too hard to achieve.”. Finally, Draper (1999) argues that the element of fun in interactive systems as a means to achieve learning and learnability goals. In this work, the concept of flow is identified as one of the methods to achieve fun.

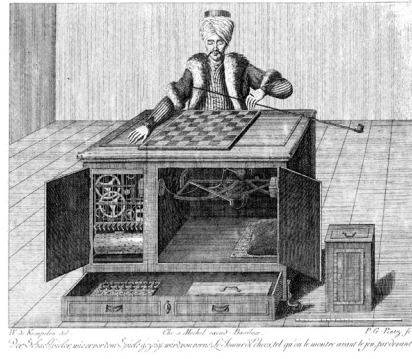
### I.3 SOCIAL ROBOTICS – AN EMERGING TECHNOLOGY

The word “robot” was first coined by Czech writer Karel Čapek in his 1920 science fiction play *Rossum's Universal Robots*. The word he used for his mechanical automatons is derived from the word *robota*, which is Czech for “forced labor”. Of course, the idea of mechanical men is much older than Čapek’s play. The idea however is much, much older than the technology to actually create such machines. The ancient Greeks had myths about the

mechanical servants of Hephaestus, the smithing god, Leonardo da Vinci created plans for mechanical humanoids in 1495 (fig. 1.13), and in the 18<sup>th</sup> century, the Mechanical Turk (fig. 1.14), a chess-playing automaton, captured the imagination of people worldwide, though this turned out to be mechanical illusion controlled by a person inside the device. To this day, fiction continues to shape the robotics conversation, even though the capabilities of robots in fiction are still far ahead of current technology. The portrayal of robots in fiction is diverse; archetypes range anywhere from evil killer robots such as the Terminator, to sympathetic companions such as Star War's R2D2 and C3PO, which continues to shape the cultural perception of robots (Sundar et al., 2016).



**Fig. 1.13** Model of da Vinci's mechanical knight<sup>19</sup>.



**Fig. 1.14** The Mechanical Turk. Adopted from Windisch (1783).

The word robot is a very broad term, and means different things to different people. Consequently, it is difficult to define what properties constitute a robot exactly, and definitions are quite diverse. The Oxford dictionary describes robots as “A machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer”. On the other hand, the Robot Institute of America offers a more strict definition: “A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of task”.

As evidenced, the word “robot” can be interpreted quite broadly, though for our purposes, we wish to emphasize two aspects: (1) a robot is a physical thing that interacts with the real world, (2) a robot be reprogrammed or reconfigured to fulfill different tasks. Such a definition precludes completely virtual agents, such as search engine web crawlers, which are sometimes also called robots. It also precludes most machines in everyday life, such as a household appliance or a vending machine. These electromechanical machines interact with the physical world to perform a task, but they cannot be reconfigured to perform anything outside of that one task.

<sup>19</sup>Photo by Erik Möller. Leonardo da Vinci. Mensch–Erfinder–Genie exhibit, Berlin 2005. Public domain.

In recent years, robotics technology has begun to attract considerable attention. Already in 2007, computer visionary Bill Gates noted parallels between the rise of personal computers and the evolution of robotics technology (Gates, 2007):

*“As I look at the trends that are now starting to converge, I can envision a future in which robotic devices will become a nearly ubiquitous part of our day-to-day lives. I believe that technologies such as distributed computing, voice and visual recognition, and wireless broadband connectivity will open the door to a new generation of autonomous devices that enable computers to perform tasks in the physical world on our behalf. We may be on the verge of a new era, when the PC will get up off the desktop and allow us to see, hear, touch and manipulate objects in places where we are not physically present.”*

Gates argues that the decreasing costs of hardware, sensors and computer processing power will make robotics more and more ubiquitous. In the 1970s, no one could have predicted the far-reaching impact of personal computers on society. Similarly, no one can truly predict the shape of a society where robots are commonplace.



**Fig. 1.15** Examples of commercial Internet of Things (IoT) products. Starting from the top left and moving clockwise, they are: Nest thermostat, Google Home, Amazon Dash, Nike Fuel, Philips Hue, Apple watch.

In the past five years, there has been a tremendous push toward ubiquitous technology. Because of trends such as IoT, smart devices and wearable devices, we are never more than a few meters removed from computers. Products like the Nest thermostat, Amazon Dash, and the Apple watch (shown in fig. 1.15) exemplify this. Speech recognition and synthesis

technology is steadily reaching a point of maturity, as illustrated by voice-driven products from companies such as Google, Amazon and Apple. Furthermore, the first practical applications of Artificial Intelligence (AI) and machine learning have started to appear across a wide range of consumer products and services, including autonomous vehicles (Google, Tesla), conversational agents (Google Assistant, Amazon Echo), automatic media curation (Netflix, Spotify), and even artistic expression (Prisma).

As technology evolves, so must our interaction with it. Nowadays, large parts of our lives are driven by the internet. However, there is still a dichotomy between the digital world and the physical world; and for the foreseeable time, we will remain denizens of the physical world. As of yet, screens and keyboards and mice remain the predominant interface between these two worlds. Surely we can come up with better modes of interaction? Even with recent developments in speech technologies and conversational AI, one cannot help but ponder the bizarreness of conversing with a small, inanimate box.

Several scholars advocate a more holistic approach toward bringing the physical and digital worlds together. Ishii (2008) argues in favor of moving beyond graphical user interfaces toward *tangible* user interfaces (TUIs). In what he calls tangible bits, users employ all their senses and abilities to directly interact with a tangible representation of an underlying data model. One way of implementing tangible bits is by using a robotic agent as an interface, leading to the concept of a Robotic User Interface (RUI). This metaphor is especially useful when there is a strong social or emotional dimension to the interaction. And as Bartneck, Reichenbach, et al. (2004) remark, one aspect sets robotic agents apart from virtual ones: they live in the physical world and can interact with it accordingly. As the authors summarize: “Screen characters simply cannot bring you a cup of tea”.

As of yet, robots are designed by a small and narrow group of people. However, we believe the field of robotics stands much to gain from a broad group of users that are not expert roboticists. These users are not burdened by the same preconceptions, and can offer better insight into context, usage and applications of social robots. Taking into account the complexity of robots and differing perspectives of each stakeholder, we believe in an evolution to a continuous spectrum of robot designers, similar to Fischer’s consumer/designer spectrum (section ). In such a spectrum, technology experts take up the role of meta designers, focussing on creating tools for other designers to use. Similar to how the Arduino platform expanded microcontroller technology beyond the audience of electrical engineers, we wish to open up social robotics technology to an audience beyond that of roboticists. Doing so is bound to lead to novel, valuable and unexpected results.

### 1.3.1 CLASSIFICATION

One of the first large-scale applications of robots was in industrial automation. The first commercially available industrial robot was the Unimate, which was introduced in the late 1950s. In 1961, the robot was first put to use in a General Motors factory, where it performed the task of transferring hot metal castings into a cooling bath (Nof, 1999). Even today, industrial robots have an unmistakable role in manufacture, and especially in



the automotive industry.

The automotive industry remains the most important customer of industrial robots; the International Federation of Robotics estimates that 38% of the 254,000 units sold in 2015 are used in automotive. In terms of units, it is estimated that the number of industrial robots worldwide will continue to grow: from 1,631,600 operational industrial robots at the end of 2015 to 2,589,000 units at the end of 2019. (IFR, 2016a)

Traditionally, robots are used for tasks that fulfill the three D's of robotics: dirty, dangerous, and dull. They would perform repetitive and dangerous tasks that were unsuitable for human workers. Nowadays, robots are growing more and more beyond this strict role, moving closer and closer to human living spaces. As Bill Gates (2007) hints at, there are remarkable parallels between the historical evolution of computers, and the current trends in robotics.

The first digital computers took the form of massive, building-size mainframes that only the largest of companies could afford. These shared computers were accessed through remote terminals. As computer technology advanced, we saw the introduction of *personal* computers; computers that were small enough to put on a desk, that were affordable for regular consumers. Nowadays, technology has begun to move toward ubiquitous computing, where advanced computers are available nearly anywhere.

Presently, we see that the evolution of robotics has begun to follow a similar course. The first robots were big, bulky and dangerous machines positioned in assembly lines. Nevertheless, with each new generation of robots, the field of robotics moves further beyond the factory floor and closer to people. Even within manufacture environments, people have begun to think differently about the role of robots. For instance, Takayama et al. (2008) suggest that it may be opportune to let robots work together with humans, instead of replacing them.

This shift in thinking is made possible by advancements in robotics technology, and can be summarized as follows: human society should not be adapted to accommodate robots, rather robotic technology should be adapted to integrate with human society. In the past, the environments of industrial robots were optimized to suit them: they are put in cages so they cannot hurt anyone. Nowadays, robots are starting to get introduced in human environments, where they are expected to deal with environment variables in a safe, predictable manner. One commercial robot that embodies this approach is Baxter (fig. 1.17), a safe, easy-to-program robotic arm designed to work in close proximity with humans (Guizzo and Ackerman, 2012).

Following this reasoning, we can discern three categories of robots, as shown in figure 1.16. Each successive group moves closer to the living space of humans. The categories are:

1. *Industrial robots* – ISO 8373:2012 offers the following definition:

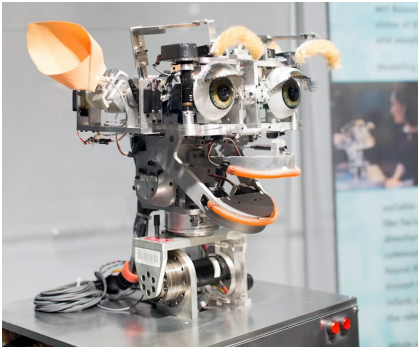
*“An industrial robot is an automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be*



**Fig. 1.16** Three types of robots: (A) industrial robots, e.g. KUKA robot arm, (B) service robots, e.g. Savioke Relay, (C) social robots, e.g. Jibo.



**Fig. 1.17** Baxter.



**Fig. 1.18** Kismet.

*either fixed in place or mobile for use in industrial automation applications.”*

An example of an industrial robot is a robot arm that welds together car chassis parts.

2. *Service robots* – ISO 8373:2012 defines service robots as follows:

*“A service robot is a robot that performs useful tasks for humans or equipment excluding industrial automation application. Note: The classification of a robot into industrial robot or service robot is done according to its intended application.”*

Within service robots, the norm makes a distinction between professional service robots and personal service robots. The former serve a commercial task, and is

usually operated by a professional. For instance, an ROV that is used to inspect underwater oil fields. The latter fulfill a non-commercial task, and are generally used by lay persons. An example of a personal service robot is the Roomba vacuum-cleaning robot.

3. *Social robots* – Bartneck and Forlizzi (2004) give the following definition:

*“A social robot is an autonomous or semi-autonomous robot that interacts and communicates with humans by following the behavioral norms expected by the people with whom the robot is intended to interact.”*

One of the first social robots was Kismet (fig. 1.18), a robotic head created by Breazeal (2003a).

The International Federation of Robotics also reported a considerable rise in commercial service robot sales. Approximately 41,060 professional service robots were sold in 2015 (a 25% increase). The two main markets for professional service robots are logistics and defense. In the same period, nearly 5.4 million personal service robots were sold (a 16% increase). Personal service robots sold in 2015 mainly fulfill household tasks (e.g. vacuum cleaning, lawn mowing) and entertainment purposes. Nevertheless, the report predicts a strong increase in sales of robotic companions and assistants in the period 2016-2019. (IFR, 2016b)

Breazeal (2003c) describes four classes of social robots, based on their ability to support the social model in more complex environment and for more complex scenarios:

1. *Socially evocative* – Artifacts that encourage users to anthropomorphize the technology when interacting with it. Commonly used in toys, such as the Tamagotchi.
2. *Social interface* – Robots that facilitate interactions with people by leveraging human-like social cues and communication modalities. Social behavior is superficial, leading to a shallow and reflexive social model.
3. *Socially receptive* – Robots that benefit from interaction with humans, for example by learning through imitation. They are socially passive, responding to social cues but not pro-actively pursuing social interaction.
4. *Sociable* – Social “creatures” that actively engage in interaction in order to satisfy internal goals and motivations. They do not only perceive social cues, but understand and interpret them on a deeper level.

Fong et al. (2003) complements this classification with three additional categories:

1. *Socially situated* – “Robots that are surrounded by a social environment that they perceive and react to. Socially situated robots must be able to distinguish between other social agents and various objects in the environment.”

2. *Socially embedded* – “Robots that are: (a) situated in a social environment and interact with other agents and humans; (b) structurally coupled with their social environment; and (c) at least partially aware of human interactional structures (e.g., turn-taking).”
3. *Socially intelligent* – “Socially intelligent. Robots that show aspects of human style social intelligence, based on deep models of human cognition and social competence.”

With this work, we focus on the basal layers of this classification; aiming to create *socially evocative* and *social interface* robots. Our reasoning for this choice is two-fold. To begin, higher levels of social interaction are primarily the result of the robot’s behavior, i.e. its programming. On the other hand, our work is primarily concerned with the embodiment design of social robots. We wish to create a system that has the affordances to support social interaction with humans; e.g. facial expressions, gaze, speech... Secondly, as indicated by the IFR report on service robots (IFR, 2016b), social robots have the potential of being used in a very wide range of applications. With our work, we wish to offer the low-level building blocks to address this broad application scope.

### 1.3.2 EMBODIMENT AND APPEARANCE

Social interaction between humans and robots is at this point the subject of much scientific research. This is perhaps unsurprising, as technology – both software and hardware – has evolved to the point where practical applications of robots in daily life become more feasible. One of the core concepts enabling the field of HRI is the idea that we humans have a tendency to recognize and project emotions onto objects, requiring only the smallest hints of facial features or human-like behavior. Indeed, social interaction with robots works because humans recognize a part of themselves in robots, and robots without any socially expressive features are seen as cold or distant (Bartneck, Reichenbach, et al., 2004). Duffy states that humans’ propensity to anthropomorphize is not seen as a hindrance to social robot development, but rather a useful mechanism that requires further examination and employment in social robot research (Duffy, 2003). In the process of robot-human communication, the robot’s face plays a vital role (Disalvo et al., 2002), facial expressions are a natural means of expressing emotions towards humans. Beyond that, a social robot should possess a character and personality, noticeable by humans (Kedzierski et al., 2013). Naturally, different target applications for social robots impose a different set of requirements on the personality and embodiment of the robot. Research has shown that the embodiment of a robot has a far-reaching impact on the way that robot is perceived by humans (Wainer et al., 2006). Goetz et al. (2003) show that a robot’s appearance and demeanor can meaningfully impact the user’s willingness to cooperate with a robot. Li et al. (2010) demonstrate the effect of appearance on the likability of a robot. Generally, it is also accepted that physical embodiment enhances a robot’s social presence (Leite, Pereira, et al., 2008; Wainer et al., 2006), and that tactile interaction is a key mode of interaction in HRI scenarios (Lee et al., 2006). Bartneck, Reichenbach, et al. (2004) argue that while physical embodiments are not better than virtual agents at expressing emotions, virtual

agents do not have the same ability to interact with the physical world. For instance, the robot Travis (Hoffman 2012, fig. 1.19) exploits this property to incorporate a smartphone in the embodiment design in a way that is meaningful for social interaction.

The perception of a robot by humans is determined by two main factors: (1) the robot's appearance and (2) the robot's behavior (Goetz et al., 2003). A robot's appearance creates assumptions about the robot's capabilities. The robot's appearance should match its task and its capabilities (Bartneck and Forlizzi, 2004). For instance, a humanoid robot is usually expected to have speech capabilities (Bartneck and Forlizzi, 2004). The appearance of a robot is important because it directly influences the user's expectations about the robot's behavior and mental state, and because human-robot interaction is enhanced by an attractive or interesting appearance (Disalvo et al., 2002; Leite, Martinho, et al., 2013). According to Bartneck and Forlizzi (2004), the shape, size and material qualities of a social robot should match the task it is designed for in order to avoid false expectations. Research also suggests that there is a cultural component to the effects of a robot's appearance, and that robot designers should take cultural differences into account (Li et al., 2010).



**Fig. 1.19** Travis.



**Fig. 1.20** Nao.

While some work is being done to explore the effects of embodiment design in human-robot interaction (e.g. De Beir, H.-l. Cao, et al. 2015; De Beir and Vanderborght 2016; Hoffman 2012; Westlund et al. 2016), most experiments consider the appearance of the robot as an external constraint, focusing experimentation on other aspects of interaction. The current state of the field is that many different studies are done with the same robots (e.g. Nao; Gouaillier et al. 2008, fig. 1.20). This is understandable considering the downsides of building custom robots for an experiment, such as the monetary cost, the time effort, and the robustness and reliability of new prototype robots.

Researchers are often faced with the choice to either design their own social robot from scratch or to use/buy a commercial robot. The first option affords a large degree of flexibility, allowing researchers to fine-tune the embodiment of the robot to the exact specifications of their experiment, at the expense of development time and money. The second

option allows for a much faster, less expensive development cycle and is often preferred in research contexts. However, this speed comes at another cost: customization is often limited to software and superficial embodiment changes (e.g. giving a Nao robot eyebrows; De Beir, H.-l. Cao, et al. 2015). This dichotomy is also reflected by Tilden’s argument (2013) that the greatest barrier for breakthroughs in personal robotics is cost, in terms of both money and time. Researchers need better tools to easily and rapidly design different embodiments for social robots in order to progress the insights in HRI. Already in 2003, Fong et al. stated that most research in HRI has not yet explicitly focused on design of embodiments and much research remains to be performed. Unfortunately, few tools are available to rapidly prototype social robot embodiments, even though evidence shows that embodiment and appearance are of key importance during the interaction with humans.

With respect to robot morphology, Fong et al. (2003) describes a dichotomy between *biologically inspired* vs. *functionally designed* robots. Yanco and Drury (2004) extends this classification to three categories of social robot morphologies:

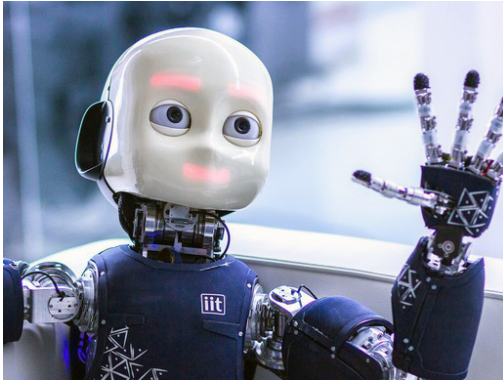
- *Anthropomorphic* – Robots with a human-like appearance.
- *Zoomorphic* – Robots with an animal-like appearance.
- *Functional* – Robots of which the appearance is predominantly determined by its function.

Similarly, Bartneck and Forlizzi (2004) describes a robot’s form as a spectrum ranging from abstract, to biomorphic, to anthropomorphic. Finally, Duffy (2003) describes a three-axis design space for the anthropomorphism of robot heads. This model is based on three morphology extremes: *human-like*, *abstract*, and *iconic*.

The role of the physical embodiment of a social robot plays a key role within HRI. If we want to investigate these interactions, we need new tools to be able to change the robot’s embodiment to the nature of the experiment. With our work, we offer a prototyping tool to explore different embodiments for social robots, with an emphasis on face-to-face communication via anthropomorphic robot heads.

### 1.3.3 DIY AND OPEN SOURCE IN ROBOTICS RESEARCH

As mentioned in section 1.1, the open source paradigm has also had a measurable impact on robotics research in general. Prominent examples of open source robotics software include the operating system ROS (Quigley, Conley, et al., 2009) and the simulation tool Gazebo (Koenig and Howard, 2004). In recent years we have also seen a rise in open source robot hardware, covering various embodiment archetypes. Examples include humanoids iCub (Tsagarakis et al., 2007) and Poppy (Lapeyre et al., 2014), the quadruped Oncilla (Sproewitz et al., 2011), and the OpenArm (Quigley, Asbeck, et al., 2011) robotic arm. Outside of these complete platforms, some hardware projects implement only one specific



**Fig. 1.21** iCub.



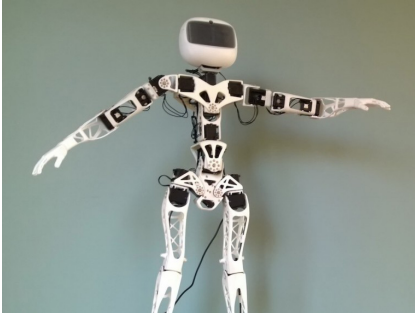
**Fig. 1.22** Hacked MyKeepon. Adopted from M. Michalowski et al. (2013).

element, which is intended to be integrated in a robotic system. Examples include the OpenHand manipulator (Ma et al., 2013) and TakkTile sensor (Tenzer et al., 2014).

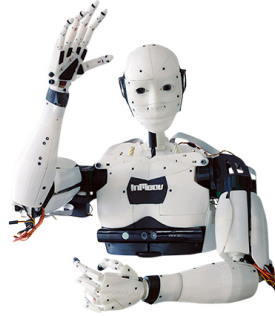
The iCub humanoid robot (Tsagarakis et al. 2007, fig. 1.21) is a prominent example of an open source social robot. With 53 degrees of freedom, the robot features a fully articulated body. The iCub also has some facilities for facial expressions: it has physically actuated eyes and eyelids, as well as eyebrows and a mouth realized using an LED matrix. Although the robot's hardware is open source, we argue that the iCub is not necessarily DIY appropriate. The robot sports expensive components including high-end sensors and motors. Moreover, the robot contains many CNC-milled as well as moulded plastic parts, necessitating a very well equipped lab to copy the design. With a retail price of 250,000 €, the platform is out of reach for all but the largest of research institutes. All in all, these design aspects make it difficult for amateurs to reproduce or modify the iCub.

The Keepon, shown in figure 1.22, is a small, abstract robotic creature (Kozima, Nakagawa, et al., 2005). With only four Degree of Freedoms (DOFs), it certainly is less embodied than the iCub. Nevertheless, the Keepon design accomplishes a large degree of lifelike behavior and emotional expressiveness with a very limited set of actuators. The original, research-grade Keepon Pro is sold at a price of \$30,000 (H. L. Cao et al., 2014). A simplified toy version called MyKeepon was released in 2010, and costs only \$40. This toy version has enjoyed some attention from both DIYers and HRI practitioners: the toy can be hacked using an inexpensive Arduino board, enabling manual control of the robot's systems (H. L. Cao et al., 2014). Another example of this approach of hacking commercial systems is given by De Beir, H.-l. Cao, et al. (2015); they increased the expressive range of a Nao robot by retrofitting actuated eyebrows to the robot's face.

More recently, new platforms have emerged that place a more explicit emphasis on community-driven modifications and development. For instance, the Poppy project (Lapeyre et al., 2014) focuses on robot designs based on 3D printed components in conjunction with



**Fig. 1.23** Poppy Humanoid.



**Fig. 1.24** InMoov.

Dynamixel-brand smart servos. The use of 3D printing enables quick and accurate reproduction of parts, but also allows the designs to be altered quickly. Currently, the project offers three designs: a 6 DOF arm, a 13 DOF upper torso, and a 25 DOF humanoid robot (fig. 1.23). Optionally, an LCD screen can be added to the head of the robot, enabling the robot to display basic facial expressions.

Another example is the InMoov robot<sup>20</sup>, shown in figure 1.24. InMoov is an open source, 3D printed humanoid torso created by a French sculptor and designer. The robot bears many similarities Poppy, though the InMoov is life-size whereas the Poppy humanoid is scaled down. The robots are also built using different technologies: InMoov uses hobby servos combined with low-cost FDM printed parts, whereas Poppy uses more expensive Dynamixel servos combined with selective laser sintered parts. This difference is reflected in the cost: InMoov is estimated to cost around \$900 (Fornari and Cangiano, 2015), whereas the Poppy torso is sold for 5300 €.

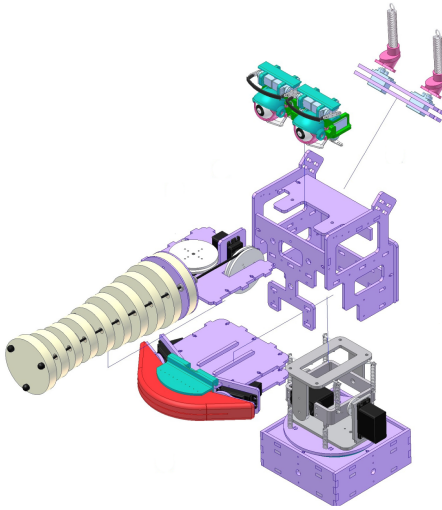
Figure 1.25 shows a recently redesigned version of social robot Probo. Originally, Probo was developed as an advanced research platform for HRI studies with children, with a primary focus on face-to-face communication. As described by Goris et al. (2011), the robot is constructed from CNC milled aluminium and 3D printed plastic parts. Movement is accomplished through compliant cable-driven actuators. For the redesigned version, a different approach was used. Doroftei et al. (2016) cite complex construction and expensive actuators as the primary reasons as to why Probo is difficult to reproduce. Therefore, a new version of the head was designed based on DIY-appropriate techniques. As illustrated in figure 1.25, the second version uses low-cost hobby servos mounted to a laser-cut plastic frame.

Finally, some HRI practitioners are experimenting with affective communication through robots built using LEGO Mindstorms. One of the earliest examples embodying this approach is Felix, a four-DOF robotic face built using the original LEGO Mindstorms kit (Cañamero and Fredslund, 2000). Since then, two descendants were built Seymour and Moodles<sup>21</sup>, shown in figure 1.26. Others have since followed with Mindstorms robots

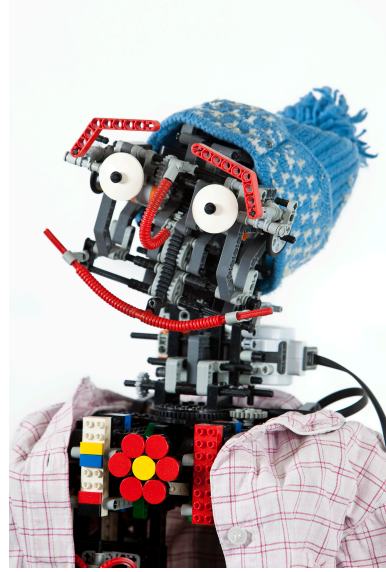
<sup>20</sup>InMoov: open-source 3D printed life-size robot – <http://inmoov.fr>

<sup>21</sup>Felix homepage – [http://www.cs.au.dk/~chili/feelix/feelix\\_home.htm](http://www.cs.au.dk/~chili/feelix/feelix_home.htm)





**Fig. I.25** Redesigned Probo head. Adopted from Doroftei et al. (2016).



**Fig. I.26** Moodles.

that communicate emotion through facial features (Kipp and Schneider, 2016) or body motion (Novikova and Watts, 2015). While using Mindstorms is a cheap and easy method to construct robotic faces, the tool is not designed for this purpose. As a result, the robots tend to have a mechanical appearance with limited degrees of freedom.

Presently, researchers are often faced with the choice to either design their own social robot from scratch or to use/buy a commercial robot. The first option affords a large degree of flexibility, allowing researchers to fine-tune the embodiment of the robot to the exact specifications of their experiment, at the expense of development time and money. The second option allows for a much faster, less expensive development cycle and is often preferred in research contexts. However, this speed comes at another cost: customization is often limited to software and superficial embodiment changes.

As of yet, little progress has been made toward toolkits for human-robot interaction. Most existing modular humanoid robots, such as Poppy (Lapeyre et al., 2014) or DARwIn-OP (Ha et al., 2011), focusing on body motion, with limited room for the exploration of different embodiments and leaving the face virtual or static. Researchers need better tools to easily and rapidly design different embodiments for social robots in order to progress the insights in HRI. To address the current difficulties of designing custom social robots, we identify an open source, DIY-friendly toolkit approach as one potential solution.

## I.4 RESEARCH QUESTION AND DESIGN GOALS

In the previous sections, we have attempted to sketch an overview of the relevant contemporary technology trends. From the various influences, we recognize an opportunity for an open DIY platform that facilitates the design, construction, and production of new social robot characters. With the work described in this thesis, we address this need. This leads to the central research question of this work: *“How can we enable non-experts to design, build, and use custom social robots?”*

From the works referenced in the preceding sections, we crystallize a number of design goals, setting the stage for the rest of this work. The design goals form a starting point to tackle the central research question. These goals are further explored using a *research through design* methodology, where concrete case studies are used to investigate the design context. This methodology is described in detail in the next chapter. The combination of these two approaches enables us to investigate the complexities of the research context in an integrated, holistic manner. In this work, we identify and target the following design goals:

- *Open ( $G_1$ )* – The system should be designed so that users are free to build it, hack it, and use it in any way they see fit. The open source paradigm offers a good solution to enable this behavior. However, making source files available is only part of the answer. The platform should take into account the barriers that stand in the way of replication and modification. These barriers include aspects such as skill, cost, and infrastructure. Design decisions should take these aspects into account; certain components or techniques (e.g. CNC milling) might lead to a better performing design, but are less accessible to amateurs, thereby hampering reproduction and adaptation.
- *Easy to build ( $G_2$ )* – It is essential that the hardware is designed so that it can be built and modified by non-expert users. By building the system themselves, users become more experienced in the design and functioning of the robot. This experience will also make users more confident in repairing and modifying the robot. By reducing the knowledge and skill requirements, we ultimately enable a large audience of users to create their own robots.
- *Emotionally expressive ( $G_3$ )* – The toolkit should let users construct robots with affordances for social expression, including facial expressions, speech, and gaze. The software should be able to automatically generate appropriate facial expressions from emotions. Users need to be able to control the emotional expressions through simple interfaces.
- *Facial features ( $G_4$ )* – Within this work, we focus on design of the robots that are socially expressive through face-to-face communication. Robots that rely on body gestures rather than facial features to communicate emotion are outside of the scope of this work.

- *Character* ( $G_5$ ) – Robots built using the toolkit must not look like disembodied, mechanical robot heads. Their embodiments should resemble artificial characters rather than robots. Using the toolkit, it must be possible to create biologically-inspired characters with a socially evocative appearance.
- *Creativity* ( $G_7$ ) – The platform should inspire the user to be creative. Everyone has the potential to be creative, though not everyone has the same level of creativity nor the same desire to create. The system should have a low threshold to go from consuming to creating, and the system should offer multiple ways of expressing creativity, ranging from quick and easy to deep and fulfilling.
- *Flow* ( $G_6$ ) – As a way of stimulating creativity, the platform should be designed to stimulate the user to enter a mental state of flow. This state can be reached when the challenges of an activity are tuned to the skills of a user. In this state, users have the feeling of being “in the zone”. Flow is a powerful way to stimulate fun and learning while using complex interactive systems.
- *Diverse knowledge domains* ( $G_8$ ) – Robotics is interdisciplinary by nature. However, as of yet, emphasis is placed on engineering and technology aspects. Our toolkit needs to incorporate aspects from a wider range of disciplines in order to reach a wide audience. In addition to the traditional robotics knowledge domains, the toolkit needs to incorporate elements from crafts, design, and art.
- *Low cost* ( $G_9$ ) – Cost forms an important barrier in the adoption of robotics technology. Consequently, attention should be paid to the monetary cost of the system. Many social robots are only affordable for large universities and research centers. Even then, cost often hampers large-scale experimentation and wider adoption. Component cost also poses a barrier for replication and modification, thereby hampering the evolutionary process that drives open source projects. Additionally, low-cost social robots provide advantages that current high-end robots do not have: they are well suited for large-scale studies and they are accessible to a wider audience.
- *Community-oriented* ( $G_{10}$ ) – The platform should rely on existing projects from the DIY community where appropriate. The platform itself should be designed in a way that allows a community of third-party users to form. We want the platform to become a true, self-sustaining, open source hardware project. Having a community around the platform offers many advantages, such as development contributions from third parties and deeper insights into actual usage scenarios of the platform.

The goals are summarized and numbered in table 1.1. This numbering is used throughout this work to refer to each specific goal. Consolidating these different goals is not always straightforward, as the combination of the different goals can sometimes lead to contradictions and paradoxes. Chapter 2 will detail why these contradictions exist, and how conflicting goals are reconciled through a user-centered design methodology.

Design goal		DIY	Social robotics	Learning
$G_1$	Open	✓		
$G_2$	Easy to build	✓		
$G_3$	Emotionally expressive		✓	
$G_4$	Facial features		✓	
$G_5$	Character		✓	
$G_7$	Creativity			✓
$G_6$	Flow			✓
$G_8$	Diverse knowledge domains			✓
$G_9$	Low cost	✓		✓
$G_{10}$	Community-friendly	✓		✓

**Table 1.1** Summary of the design goals

## 1.5 OUTLINE

This work summarizes our study on the creation of open source robotics in academia with the social robot Ono and the Opsoro design toolkit for social robots. We detail our design approach, leveraging DIY-friendly techniques to create systems that, though complex, can be assembled and modified by novices. And we describe experiments where different types of users use our work to bring new social robot concepts to life. Throughout this work, the trends and influences mentioned earlier are integrated step by step, and we detail how this has led to the creation of the Opsoro design toolkit for social robots. The rest of this work is organized according to the following structure:

- **CHAPTER 2** details the design methodology used during this project. The chapter begins by illustrating why the design of a robotic toolkit is difficult and complex from a system’s perspective. We compare the lifecycle phases and the product stakeholders of conventional products and platform-based toolkit products. This overarching model is used to argue as to why a User-Centered Design (UCD) methodology must be used. Our methodology is framed within UCD models and practices, including design thinking, usability engineering, and user experience design. We then approach the design methodology from a more practical, zoomed-in perspective. We justify why iterative prototyping is used, detail how digital manufacturing techniques are used to support iterative prototyping, and conclude with design strategies that leverage the intrinsic properties of these techniques.
- **CHAPTER 3** describes the iterative design process in detail and demonstrates how the platform was used to create different types of social robots. The chapter begins with a high-level overview of the design iterations. The chapter distinguishes five interrelated project branches: hexapod robots, Ono generation 1, Ono generation 2, Robot Blocks, and Opsoro. The chapter’s introduction positions these projects

with respect to one another, illustrating the historical evolution and the interconnectedness between the different projects. From there on, the chapter discusses each project in detail, including motivation for the design decisions as well as the experimental data that substantiates these decisions. The iterations consist of both technical iterations and user-driven iterations, in accordance with the ISO model of human-centered design of interactive systems. The experiments cover the *design*, *build*, and *use* phases of the toolkit with different types of potential users. All the design requirements discovered through the iterative, human-centered approach were implemented in the latest version of the Opsoro toolkit, which is discussed in the next chapter.

- **CHAPTER 4** gives in-depth details on the technical implementation of the Opsoro platform. The chapter discusses the mechanical design of the system, the electrical design, and the software architecture. The section on the mechanical design of Opsoro starts with an overview of the various modules for facial expressions, including the mouth, the eyebrow, the eye, the joint, and the neck. It then describes our proposed methodology for designing custom embodiments by combining the aforementioned modules with a custom design made using digital manufacturing techniques. The methodology describes the steps to design a laser-cut skeleton frame, then continues with the steps to create a soft, huggable outer skin, and concludes with a brief overview of variants on the embodiment methodology.

The second part of the chapter discusses the design of the robot's electronics. Here, the technical iterations of the main controller board are described, starting from an amalgam of off-the-shelf boards, continuing with various PCB prototypes, and concluding with a production-ready add-on board for the Raspberry Pi. This section ends with background information on the various subsystems of the board, such as the communications protocol, the servo driver implementation, and the capacitive touch sensor.

The third and final section of the chapter presents the various software components of the system. This section starts with a description of the chosen emotional model, as well as the facial expression algorithm that turns positions in this emotional space into servo motor actuations. Then, the section continues with a description of the platform's web-based user interface, including an explanation of how the various subsystems work together, as well as an overview of the interface design choices, and a description of the various apps that are available to end-users.

- **CHAPTER 5** concludes this work with a summary of our accomplishments, and an overview of future work. One of our most important achievements is that we have a seemingly complex piece of technology accessible to amateurs and enthusiasts worldwide. A recurring theme in our workshops is that participants often doubt their own skills and chances of success at the start, yet they always manage to successfully create a robot by the end of the day. If the general evolution of the maker and DIY movements are an indication of the future, this democratization of robotics technology will eventually open the door to novel and unexpected applications of social robots.

While performing this research in the context of industrial design, we have always

maintained close ties to industrial engineering and industrialization throughout the project. As part of this project's valorization, we are currently working on commercializing the research from this project in the form of a spin-off company. As part of these preparations, we have conducted market studies and business model studies which revealed that the maker & STEM education audience currently represents the best target market. Keeping true to the roots of this project, we will continue to experiment with the open source paradigm, as we remain convinced that the model opens up many more business opportunities than that it creates threats. Turning this research project into a proper commercial product will still require a big effort in development, industrialization, and marketing, but will be an interesting journey for everyone involved. The prospect of creating a business around this research is very exciting and holds promise for an interesting future of the Opsoro platform.

## I.6 LIST OF PUBLICATIONS

### JOURNAL ARTICLES

C. Vandeveldel, F. wyffels, B. Vanderborght, and J. Saldien (in press). “DIY Design for Social Robots”. In: *IEEE Robotics and Automation Magazine*

P. D. Conradie, C. Vandeveldel, J. De Ville, and J. Saldien (2016). “Prototyping Tangible User Interfaces: Case Study of the Collaboration between Academia and Industry”. In: *International Journal of Engineering Education* 32.2, pp. 1–12

C. Vandeveldel, F. wyffels, C. Ciocchi, B. Vanderborght, and J. Saldien (2015). “Design and evaluation of a DIY construction system for educational robot kits”. In: *International Journal of Technology and Design Education* 26.4, pp. 521–540

### PEER-REVIEWED CONFERENCE PAPERS

C. Vandeveldel and J. Saldien (2016a). “An Open Platform for the Design of Social Robot Embodiments for Face-to-Face Communication”. In: *Proceedings of the 11<sup>th</sup> International Conference on Human-Robot Interaction*. Christchurch, New Zealand, pp. 287–294

C. Vandeveldel and J. Saldien (2016b). “Demonstration of OPSORO – an Open Platform for Social Robots”. In: *Proceedings of the 11<sup>th</sup> ACM/IEEE International Conference on Human-Robot Interaction*. Christchurch, New Zealand

C. Vandeveldel, M. Vanhoucke, and J. Saldien (2015). “Prototyping social interactions with DIY animatronic creatures”. In: *Proceedings of the 9<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*

C. Vandeveldel, P. Conradie, J. De Ville, and J. Saldien (2014). “Playful interaction: designing and evaluating a tangible rhythmic musical interface”. In: *Proceedings of INTER-FACE: The Second International Conference on Live Interfaces*. Ed. by A. Sa, M. Carvalhais, and A. McLean. Lisbon, Portugal, pp. 132–141

C. Vandeveldel, J. Saldien, C. Ciocchi, and B. Vanderborght (2014). “Ono, a DIY open source platform for social robotics”. In: *Proceedings of the 8<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*. Ed. by A. Butz, S. Greenberg, S. Bakker, L. Loke, and A. De Luca

M. Vanhoucke, C. Vandeveldel, J. Ingels, G. Hamon, and J. Saldien (2014). “Serious game as educational tool for safety and prevention”. In: *Proceedings of the 2014 ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play – Workshop Participatory Design for Serious Game Design*. Toronto, Canada: ACM

C. Vandeveldel, J. Saldien, C. Ciocchi, and B. Vanderborght (2013c). “The use of social

robot ono in robot assisted therapy”. In: *International Conference on Social Robotics, Proceedings*

C. Vandevelde, J. Saldien, C. Ciocchi, and B. Vanderborght (2013a). “Overview of technologies for building robots in the classroom”. In: *Proceedings of the 4<sup>th</sup> International Conference on Robotics in Education*

## PEER-REVIEWED BOOK CHAPTERS

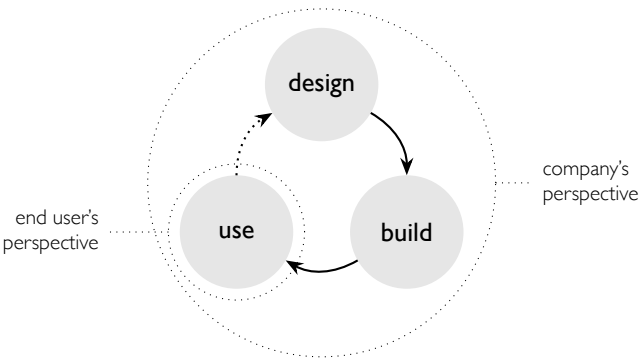
C. Vandevelde, J. Saldien, C. Ciocchi, and B. Vanderborght (2013b). “Systems overview of Ono: a DIY reproducible open source social robot”. In: *Lecture Notes in Computer Science*. Ed. by G. Herrmann, M. J. Pearson, A. Lenz, P. Bremner, A. Spiers, and L. U. Vol. 8239. Springer International Publishing, pp. 311–320



# Chapter 2

## DESIGN METHODOLOGY

The lifecycle of most products can be approximated using a relatively simple lifecycle model, shown in figure 2.1. In this model, three actions are distinguished: the design of a product, the manufacture of the product (typically through mass-production), and finally, the use of the product by a consumer. The lifecycle of such a product is relatively straightforward, and each action is performed by a different group of people. The product is designed by a company's R&D department, it is built by the people in its manufacturing department, and it is used by the company's customers. Consequently, the company's perspective encompasses the entire product lifecycle, and observations about a product's usage may influence the next generation of that product. On the other hand, the end users – consumers – are usually only interested in what value the product can offer them. How it was designed or how it was produced, is of little concern to them.

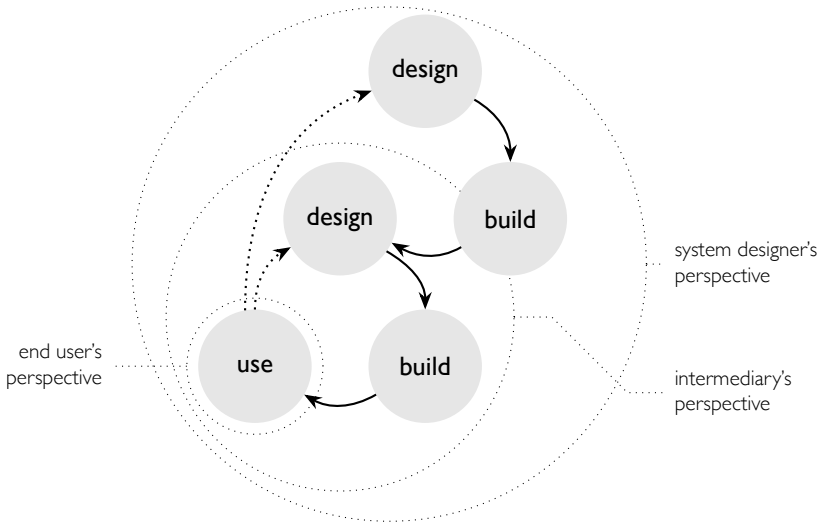


**Fig. 2.1** Lifecycle model of conventional products.

By contrast, in recent years more and more platform-based products have begun to appear. These products are characterized by a complex, double-loop lifecycle model, as illustrated in figure 2.2. One of the key characteristics is that the roles of designers, producers, and

consumers are no longer clearcut. They are instead diffused over the different product stakeholders. Toolkits (described in section 1.1.5) form an important subset within this group of products.

Unlike the first model, the *design–build–use* process is complicated by a second layer of hierarchy, as secondary *design* and *build* phases are introduced. The model distinguishes three major perspectives. The largest perspective is that of the system designers; they are concerned with the design of the toolkit, with the way the toolkit is used by others to design, and with how artifacts created using the toolkit are used. The second perspective is that of the intermediaries; the toolkit users in this case. For the intermediary, the toolkit represents a boundary condition which delimits their design space. Within this design space, the intermediary relies on their creativity to create an artifact that is relevant for end users. The intermediary can be the end user, though the end users can also be another distinct group. The final perspective is that of the end users. As with the first model, the end users' perspective is the most narrow perspective, being primarily concerned with the value of the artifact in their context. Still, characteristic of a platform-based product is that even end usage can contain elements of open-endedness and creative play.



**Fig. 2.2** Lifecycle model of platform-based products.

In this lifecycle model, there are two loops of iteration, represented by the dotted arrows in figure 2.2. The inner iteration loop happens within the context of the intermediaries. Based upon testing and feedback from end users, the intermediaries can use the toolkit to further improve the design of an artifact. The second iteration loop is situated at the system designers' level. This iteration loop is more complicated because a system designer needs to take into account both the direct users of the toolkit (the intermediaries) and the indirect users (the end users). Consequently, system designers are concerned with both how the toolkit is used and with how the end product is used. Both these activities shape the development of the system in an iterative process. Even though this indirect iteration

path is more complex, the toolkit paradigm is valuable because it is a way to bridge the information gap caused by sticky information (Hippel, 2001; Hippel and Katz, 2002).

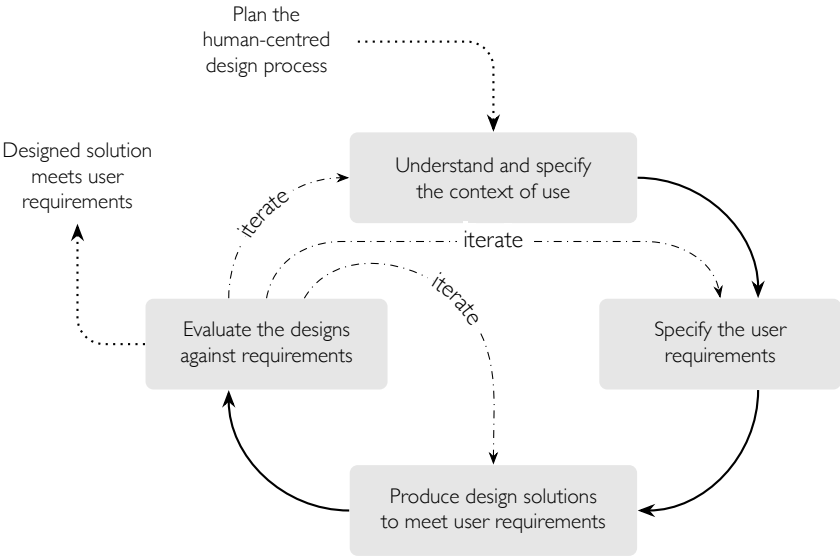
In addition to the human aspect, the context of the Opsoro platform has another degree of complexity, namely that of tangible interactivity. The system not only encompasses hardware, but also builds upon a large body of electronics and software in order to work. Our work bears many similarities to the paradigm of tangible bits, as introduced by Ishii (Ishii, 2008; Ishii and Ullmer, 1997). Ishii argues that as technology advances, it needs to move beyond screens and keyboards and permeate to all aspects of our lives. Central to the concept of tangible bits is the physical manipulation of digital data, instead of using a screen as a window into the digital world. In the form of a finished artifact, the Opsoro platform offers a different kind of bridge between the digital and physical world: that of the RUI. In its native toolkit form, components of the system do not only exist in the physical world; they have a direct counterpart in software, leading to dual citizenship of the physical and digital worlds.

In this work, we deal with distinct system activities: *design–build–use*. We deal with multiple types of designs that need to work in unison: *hardware–electronics–software*. We deal with designing a system instead of designing a product. The design of a robotics toolkit for social robots is a challenging prospect, not only from a technical point of view, but also due to the extensive influence of human factors on all aspects of the system. Therefore, we argue that conventional engineering development paradigms, such as the double diamond model (Design Council, 2005) and the waterfall model (Royce, 1970) are not applicable here, and other methods need to be used.

## 2.1 USER-CENTERED DESIGN

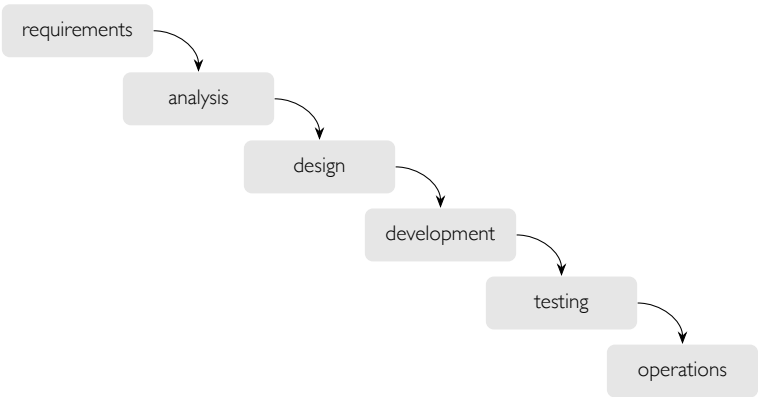
The ISO model for human-centered design of interactive systems (ISO (2010), fig. 2.3) best describes the design process of this project. It is important to note that the ISO model defines three separate iteration loops: context iteration, user requirement iteration, and design iteration. This iterative process is essential because the interaction between people and products is intrinsically hard to predict. It must be tested in situ, it cannot be divided into components that are to be tested independently in a laboratory context, and it is essential to continually question all the boundary conditions of a design.

The ISO 9241-210 model is a departure from traditional engineering design models, such as the well-known waterfall model (Royce, 1970), shown in fig. 2.4. The waterfall model is suitable for traditional engineering tasks that have fixed, well known requirements and take place in a very predictable environment. Ultimately, it is sequential in nature, and does not take into account the possibility of going backward up the cascade chain. However, being able to move back and forth between design phases is crucial in order to deal with the design challenges of interactive systems. The ISO standard summarizes the challenges of designing human-computer interaction as follows (ISO, 2010):



**Fig. 2.3** ISO 9241-210 model for human-centered design of interactive systems. Adapted from ISO (2010).

*“The complexity of human–computer interaction means that it is impossible to specify completely and accurately every detail of every aspect of the interaction at the beginning of development. Many of the needs and expectations of users and other stakeholders that will impact on the design of the interaction only emerge in the course of development, as the designers refine their understanding of the users and their tasks, and as users are better able to express their needs in response to potential solutions.”*



**Fig. 2.4** Waterfall model of development. Adapted from Royce (1970).

In practice, our experiences with the Probo project (Goris et al., 2011; Saldien, 2009) as well as the body of work described in HRI literature gave us a head start in the first step of the HCI model: understanding and specifying the context of use. Still, over the course of the project, all three iterative loops occurred multiple times: rethinking the usage context, rethinking user requirements, and rethinking the design solutions. The innermost iteration loop – rethinking the design solutions – was the most frequent iteration cycle in our case. Testing involved evaluating the appropriateness toward the intended application (HRI), as well as the appropriateness toward Maker/DIY-centric production techniques.

Our methodology incorporates many elements of *design thinking* (Brown, 2008; Dorst, 2010). This paradigm gained much traction in recent years, and offers ways to deal with problems that are common in many professions, but cannot be addressed through traditional research paradigms. Dorst frames this paradigm in terms of patterns of reasoning (fig. 2.5). Traditionally, the sciences have been predominantly occupied with deciphering the laws that govern the natural universe. The predominant reasoning patterns within the sciences are *deduction* and *induction*, leading to the well-known hypothesis-driven research paradigm:

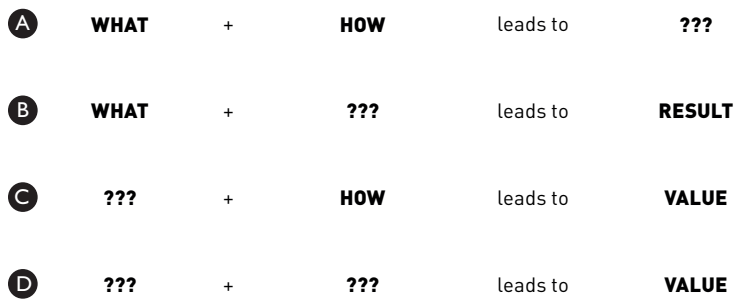
- *Deduction* – Using the initial conditions and a working principle to predict the end result of a phenomenon.
- *Induction* – Using the initial conditions and the end result to determine the working principle governing a phenomenon.

By contrast, the predominant pattern of reasoning in design professions is *abduction*. Abductive reasoning comes in two forms:

- *Abduction-1* – Using a known working principle to design a thing that create value. The thing can be an object, a service, a system. This form is often associated with problem-solving.
- *Abduction-2* – In this pattern, only the desired value is known, and there is no known working principle that guarantees value.

It should be noted that all four patterns are used in science and in design, though each discipline has a different emphasis. However, Dorst argues that design professions are fundamentally different from fields that emphasize analysis (deduction and induction) or problem solving (abduction-1), necessitating different methods and practices. Deductive reasoning guarantees the validity of the result if the premises are true. By way of contrast, abduction is the logic of “best guess” leaps, and the conclusion may turn out to be false even if the premises are true (Kolko, 2010).

The process of abductive thinking in design is further detailed by Kolko (2010). He describes design as fundamentally rooted in synthesis: “synthesis of market needs, technology trends, and business needs”. This contrasts with the sciences, where analysis is the



**Fig. 2.5** Four patterns of reasoning: (A) deduction, (B) induction, (C) abduction-I, (D) abduction-2. Adopted from Dorst (2010).

predominant mode of reasoning. Brown (2008) describes the design thinking process not as a series of orderly steps, but as a spectrum of spaces: inspiration, ideation, and implementation. Most projects visit these three spaces repeatedly as the design gradually becomes more defined. Buchanan (1992) states that design problems are indeterminate, and therefore *wicked problems*. Wicked problems are problems that are resistant to resolution because of complex, contradictory, changing requirements (C. W. Church, 1967). These problems cannot be approached without the integrative, interdisciplinary action that is inherent to design thinking.

Dorst (2010) explains that design challenges are driven by a core paradox; statements that are true in their own right, but cause conflict when combined. The core paradox leads to a field of tension; a parameter space that the designer has to explore. This leads to the importance of *reframing*: finding novel standpoints from which a problem can be approached in order to generate new solutions.

In this work, the paradox can be summarized as the desire to create high-functionality robots using simple, low-tech DIY building blocks. IN essence, our paradox is caused by the uncertainty and unpredictability of the human aspect, resulting in a facet of wickedness. It is fairly straightforward to define the functionality requirements of a service robot. A parking robot, for instance, should be able to lift a car, it should be able to navigate a parking complex, and it should not be able to harm users or damage objects. A social robot is the opposite, its functionalities cannot be predicted easily because a social robot is not a task-oriented artifact.

The question remains: where lies balance? At what point can we no longer rationalize the marginal cost of the system – in its broadest sense – by the marginal improvement. Because of the complexity of this design space, design requirements cannot be predetermined. The requirements should be discovered through a joint discourse with users. The integral aspect of human affect is a crucial differentiator between a service robot and a social robot, which is why usability and user experience are so important as a way to validate the “functionalities” of a social robot.

## 2.1.1 USABILITY & USER EXPERIENCE

The concept of usability originally arose from the software world. As computer systems grew more complex, more full-featured, and more widespread, an increasing demand appeared for these systems to be simple and efficient to operate. One of the most well-known examples of usability in software is the transition from text-based command line interfaces to graphical user interfaces. Though the concept of usability engineering was originally developed for computer applications, the same methodology was later also applied to physical products, especially those with complex functionality.

At its core, the concept of usability is task-focused. It is concerned with questions such as “*How well can a user complete a task?*” and “*How fast can a user learn to do a task?*”. The ISO 9241-11 standard defines the following usability aspects of visual display terminals (ISO, 1998):

- *Effectiveness* – A measure for the accurateness or completeness with which the user completes a goal or subgoal.
- *Efficiency* – A measure for the resources expended to reach a certain goal. Relevant resourced defined by the standard include “mental or physical effort, time, materials, or financial cost”.
- *Satisfaction* – A measure for the extent to which the users are free from discomfort.

While satisfaction is one of the factors of the norm, it should be noted that the term is interpreted in a very shallow and limited way. The standard only takes satisfaction into account insofar that it aids or hampers a user in certain task or activity.

In recent years, usability has been superseded by the concept of *user experience*. User experience offers an explanation as to why users prefer some products over others, even when a product offers less functionality at a higher price. With his book *The Design of Everyday Things*, Don Norman was one of the first to recognize and identify this shift (Norman, 2013). With the ever-decreasing cost of consumer goods and the rapid evolution of technology, user experience is one of the few remaining differentiators that can be used to distinguish a product from its competition. The original iPod, for example, is one of the first products to truly embody user experience. As Brown notes: “Great design satisfies both our needs and our desires. (...) The iPod was not the first MP3 player, but it was the first to be delightful.” (Brown, 2008). The device not only had an attractive shape and an intuitive interface, the iTunes store offered an entirely new way of buying and consuming music. The iPod does not focus on the task of playing music; it pays attention to every facet of experiencing music: discovering, buying, transferring, listening, curating, sharing.

Norman (2009) argues that good user experience design typically transcends the boundary of a product in its traditional sense. Instead products are part of a larger ecosystem, which again emphasizes the importance of system thinking and design thinking. Brown (2008) summarizes this sentiment as follows: “As more of our basic needs are met, we increasingly

expect sophisticated experiences that are emotionally satisfying and meaningful. These experiences will not be simple products. They will be complex combinations of products, services, spaces, and information.”

Norman (2002) also remarks that a person’s affective state has a deep effect on cognition. Negative valence leads to a depth-first, focused mode of thinking. Positive valence, on the other hand, broadens the thought process and stimulates creativity. Norman offers a simple analogy to illustrate this concept (Norman, 2002):

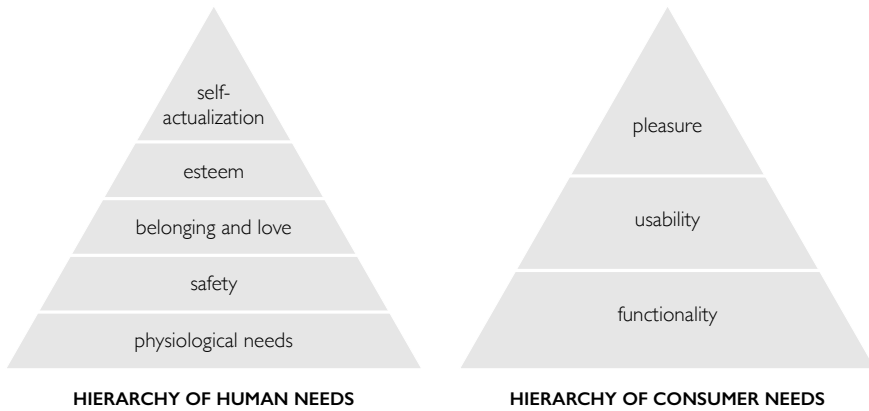
*“Imagine a plank 10 meters long and one meter wide. Place it on the ground. Can you walk on it? Of course – no problem. You can jump up and down, dance, and even walk along with your eyes shut. Now lift the plank three meters in the air. Can you walk on it? Yes, although more carefully. What if the plank were 200 meters in the air? Most of us wouldn’t dare go near it, even though the act of walking along it and maintaining balance should be no more difficult than when on the ground. Why would a simple task suddenly become so difficult – impossible, even? Tell yourself all you want that if you can walk on the plank on the ground you can also walk on it in the air. You still won’t walk along it, let alone jump and dance or, heaven forbid, close your eyes while walking. Fear dominates.”*

For this reason, UX is an important consideration in a successful toolkit. Toolkits should give end users the tools to explore a certain design space. It is important for them to be able to reframe the problems in order to create a valuable artifact. Consequently, a positive valence state of mind is necessary. In our context, a good user experience is prerequisite to positive valence, which in turn is essential to stimulate a user’s creativity (goal  $G_7$ ). In a sense, this concept in Norman’s work bears similarities to Csikszentmihalyi’s theory of flow: low valence leads to anxiety, disrupting the fragile state of flow (see section 1.2.4).

Jordan (2000) argues that usability has transitioned from a marketable feature to something that users have come to expect in a product. The presence of good usability in a product used to be a *satisfier*. Nowadays, the lack of good usability has become a *dissatisfier*. Jordan also argues that while a product that has good usability is not necessarily a pleasurable product, but that a product with bad usability is unlikely to be a pleasurable product.

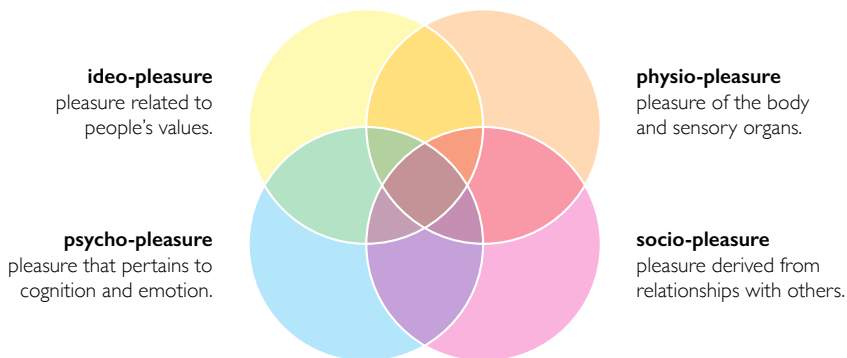
In other words, good usability is a prerequisite component of good user experience. This concept is mirrored in hierarchy of consumer needs model (fig. 2.6 right). This model is a consumer-oriented analogy of Maslow’s hierarchy of human needs (1943, fig. 2.6 left), which models human drive as a hierarchical sequence of needs. In this model, each need has the previous need as a prerequisite. The hierarchy models user experience as a chain of preconditions. At the base of the pyramid is functionality; a product which does not fulfill a function does not have a *raison d’être*, this is the primary consumer need. The secondary consumer need, usability, builds upon functionality – how easy is it to use the function of a product? The tertiary layer is pleasure (UX), a pleasurable product must always be usable and functional.





**Fig. 2.6** Hierarchy of human needs (Maslow, 1943) vs. a hierarchy of consumer needs. Adapted from Jordan (2000).

Nevertheless, Jordan argues that a usability-centric approach is inherently restrictive because it merely views products as tools to complete tasks, as opposed to objects with which humans have relations. In his model of UX, Jordan advocates a pleasure-oriented approach to product design, borrowing the four pleasures model from Tiger (1992) to classify the pleasurable aspects of products (fig. 2.7). The four pleasures model serves as a framework to organize people's characteristics so as to attain a holistic view of their requirements for a product. It should be noted that people characteristics in one category may influence requirements that pertain to a different category. For instance, physical disability, a physiological characteristic, can influence the social dimension of a product. Finally, Jordan argues that not all aspects are measurable per se. Consequently, designers should abandon the role of a scientist measuring subjects; they should instead adopt the role of a detective piecing together clues.



**Fig. 2.7** The four pleasures. Adapted from Jordan (2000) and Tiger (1992).

Hassenzahl's model (2003) identifies two dimensions of user experience: pragmatic and hedonic qualities. In broad strokes, the pragmatic qualities correspond with what is tra-

ditionally understood as usability. The hedonic qualities, on the other hand, encompass novel and hard-to-grasp concepts such as attractiveness, coolness, fun; offering an explanation as to why users prefer certain products over others. In Hassenzahl's model (fig. 2.8), product designers use their tools to convey a certain *intended* product character. Conversely, users experience a product in a certain context where the *apparent* product character ideally leads to a sense of appeal, pleasure, or satisfaction.

In addition to the pragmatic qualities of a product (which correspond with usability), Hassenzahl's model (2003) distinguishes the following three aspects of hedonic quality. They are:

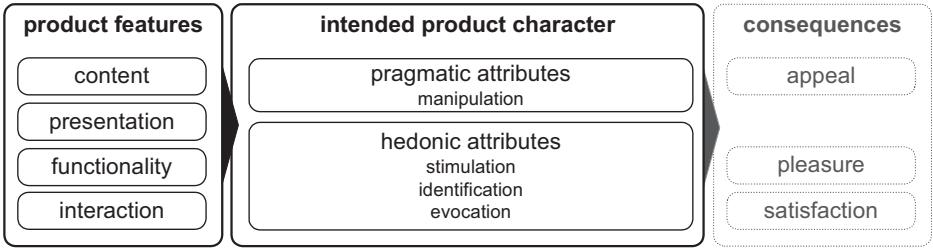
- *stimulation* – People have an innate desire for personal development, i.e. improving one's knowledge and skills. Stimulating products support this personal development.
- *identification* – People want to be perceived by others in a certain way. Products support self-expression by communicating a desired identity to others.
- *evocation* – Products that are evocative can provoke thoughts, memories, or ideas that are important to a person. Souvenirs, for instance, are products that rely on this aspect extensively.

Hassenzahl argues that the hedonic qualities and the pragmatic qualities of a product are independent from one another, leading to four distinct product characters (Hassenzahl, 2003; Hassenzahl et al., 2003). A product with weak pragmatic and weak hedonic qualities is simply unwanted and superfluous. On the other hand, a product with strong pragmatic and hedonic attributes is desirable. Hassenzahl states that “an uncompromising combination of both is the ultimate design goal.” Products with strong pragmatic and weak hedonic attributes are *act* products, and are strongly linked with a user's behavioral goals. Finally, products with weak pragmatic and strong hedonic attributes are dubbed *self* products, and are intrinsically linked with a user's identity.

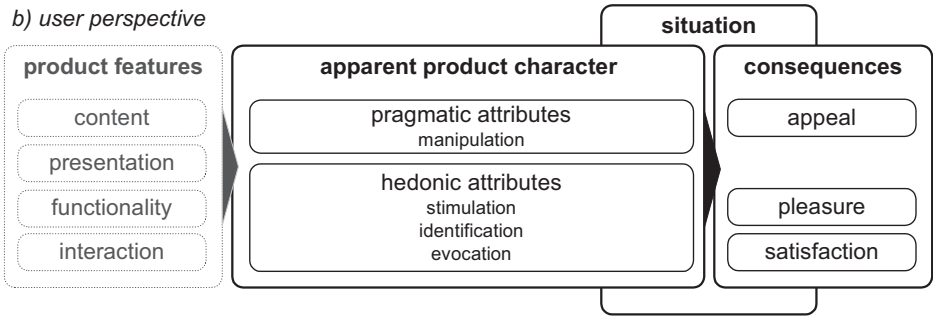
To conclude, there are clear links between user experience theory and the design goals of our work. To begin, usability aspects correlate directly to design goal  $G_2$ , easy to build. Usability also impacts goal  $G_6$ , flow. Bad toolkit usability leads to frustration and anxiety, making it impossible to enter a state of flow. A toolkit that is too easy to use will lead to boredom, so a toolkit needs the ability to grow with a user, offering new challenges as the user's skills progress.

As Jordan argues, good usability is a prerequisite for user experience design. Norman shows that user experience can directly affect a person's state of mind, which in turn has far-reaching influences on their ability to think creatively (goal  $G_7$ ). User experience aspects also impact the possibility of flow (goal  $G_6$ ). Self-directed challenges, such as elements of open-endedness and creative play, are also important for stimulating flow. These challenges are intrinsically and tightly coupled with user experience.

a) *designer perspective*



b) *user perspective*



**Fig. 2.8** Hassenzahl's model of user experience. Adopted from Hassenzahl (2003).

### 2.1.1.1 MEASURING USER EXPERIENCE

This section summarizes the measurement methods that we used to evaluate usability and user experience aspects of our prototypes. Our methodology is built upon a number of standardized UX tools, including the System Usability Scale, AttrakDiff, Pick-A-Mood, and the UX curve. These standardized tools are rigorous and well-supported in literature. However, they are generic and mostly oriented toward the user experience of software products and services, and they usually only lead to superficial insights. Moreover, user experience design is a relatively new phenomenon. Researchers and practitioners have yet to reach a consensus as to what user experience exactly entails, and what the best way to measure it is, especially for physical products (as opposed to software systems).

For these reasons, the questionnaires we use in our experiments are supplemented by custom questions. These custom questions deal with topics and issues that are specific to the experiment at hand. While less rigorous than the use of standardized tools, they offer deeper and more relevant data. In turn, this data has directly impacted the development of our platform in meaningful ways.

In our experiments, it is essential to have quick and easy-to-understand tools to evaluate user experience aspects of our prototypes. Experiments often take place in situations where participants have limited amount of time or motivation to fill in surveys. For this reason, we emphasize quick, easy to understand survey tools for quantitative data, complemented

by qualitative data from interviews, video recordings, and personal experiences.

The System Usability Scale (Brooke, 1996) uses a short questionnaire to gauge the usability of a product or system. The scale was originally developed to evaluate the usability of computer systems, but has since been used to measure products in other domains as well (Jordan, 2000). The SUS questionnaire, shown in table 2.1, consists of ten questions with five response options. Using a scoring key, the responses to these questions can be compiled into a single number between 0 and 100, which corresponds to the usability of a product or system. Care must be taken not to interpret this number as a percentage, scores should be normalized to produce a percentile ranking. Bangor et al. (2008) have produced an empirical analysis of 2324 surveys, revealing a mean SUS score of 70.14. The study also shows that the type of interface has a statistically significant influence on the mean SUS score. In summary, the SUS offers a quick way to gauge a system’s usability. However, the tool does not offer deep insight into usability, and cannot be used to identify specific issues.

		disagree				agree
		1	2	3	4	5
1.	I think that I would like to use this system frequently	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2.	I found the system unnecessarily complex	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3.	I thought the system was easy to use	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4.	I think that I would need the support of a technical person to be able to use this system	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5.	I found the various functions in this system were well integrated	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6.	I thought there was too much inconsistency in this system	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7.	I would imagine that most people would learn to use this system very quickly	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8.	I found the system very cumbersome to use	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
9.	I felt very confident using the system	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10.	I needed to learn a lot of things before I could get going with this system	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

**Table 2.1** System Usability Scale questionnaire. Adopted from Brooke (1996).

AttrakDiff (Hassenzahl et al., 2003) is a companion tool that is related to Hassenzahl’s model of UX (2003). The results from AttrakDiff surveys correspond directly to the dimensions of Hassenzahl’s model. The AttrakDiff questionnaire, shown in table 2.2, consists of a list of 28 antonym pairs (e.g. unimaginative — creative). Respondents are asked to pick a position on a seven-point scale to indicate where they believe the product is positioned between the two antonyms. Using these antonym word-pairs, the AttrakDiff tool calculates four product dimensions:

- *pragmatic quality (PQ)* – corresponds to usability, how easily can a user complete their goals with the product?
- *hedonic stimulation (HQ-S)* – to what extent does the product stimulate the user’s personal growth?

- *hedonic identification (HQ-I)* – to what extent do users identify with a product in a social context?
- *attractiveness (ATT)* – a global measure of appeal to the user.

Hassenzahl states that pragmatic (PQ) and hedonic qualities (HQ-S and HQ-I) are independent from each other, and that they contribute equally to a product's attractiveness. He also shows that pragmatic quality is correlated with task difficulty: in a study, more difficult tasks will lead to a lower perceived pragmatic quality of the product. On the other hand, the hedonic qualities are independent from task difficulty, as expected. As mentioned earlier, Hassenzahl's model (2003) also identifies a third aspect of hedonic quality: evocation. However, the AttrakDiff tool does not measure evocation as part of the questionnaire. Hassenzahl remarks that evocation only plays a subordinate role in the context of interactive products (Hassenzahl et al., 2003).

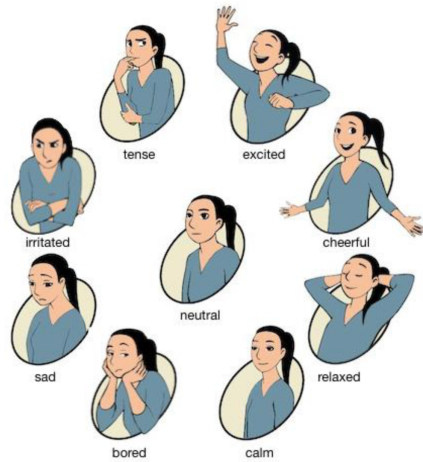
	1	2	3	4	5	6	7	
human	○	○	○	○	○	○	○	technical
isolating	○	○	○	○	○	○	○	connective
pleasant	○	○	○	○	○	○	○	unpleasant
inventive	○	○	○	○	○	○	○	conventional
simple	○	○	○	○	○	○	○	complicated
...								...

**Table 2.2** Excerpt from the AttrakDiff questionnaire. In total, 28 antonym pairs are used. Adopted from (Hassenzahl et al., 2003).

The Pick-A-Mood (Desmet et al., 2012; Vastenburg et al., 2011) tool is a very short, cartoon-based instrument that can be used to measure the mood of participants during a study. The tool comprises a set of nine cartoon drawings of different facial expressions. Users are asked to choose the drawing that best represents their mood during the course of a project, session, or workshop. Interesting to note is that the tool focuses on measuring mood, as opposed to emotion. The distinction that the authors make between the two is that emotion is momentary snapshot triggered by a single stimulus, whereas mood is the aggregate result of many different stimuli. As Desmet et al. (2012) notes, mood is a relevant parameter in design because it has a substantial impact on human behavior and social interaction, and because it can be influenced through design.

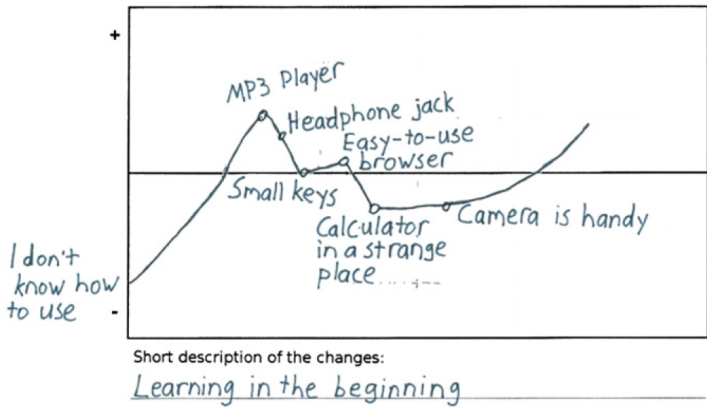
The User Experience Curve (Kujala et al., 2011) is a tool that offers a way to evaluate the evolution of user experience over time. Users are asked to draw a graph that represents how their appreciation for a product evolved, with the vertical axis representing positive/negative experiences and the horizontal axis representing time. Users are also asked to annotate key moments on the graph. An example UX curve, showing the interaction of a user with a new mobile phone, is shown in figure 2.10.

Whereas other tools summarize a user's experience in a single snap-shot measurement, the UX curve offers insight into its evolution over time. The tool is much more granular in that respect, and allows specific problem areas to be identified. Downsides of the UX



**Fig. 2.9** Pick-A-Mood expressions of eight mood types and neutral. Adopted from Desmet et al. (2012).

curve include that it has to be filled out on paper and that it takes time to explain the tool to participants, and for participants to draw and annotate their curves. Analysis of the results is subjective as the graphs need to be interpreted, though general trends can be identified by putting all graphs on top of each other digitally. Despite its disadvantages, the tool provides rich insight into user experience, and results can be further augmented using open survey questions or follow-up interviews.



**Fig. 2.10** Example of a user experience curve. The graph depicts one user's experience with a product in function of time. Adopted from Kujala et al. (2011).

In addition to the standardized tools mentioned above, custom instruments were also used in experiments. This way, insight could be gained into specific aspects of usability and user experience of a certain prototype. Still, the custom instruments are subject to the same constraints as the standardized surveys, namely: they should be quick and easy to use.

To begin, most surveys we used in experiments include custom Likert scale questions. These questions can be filled in quickly by participants, and allow us to gauge very specific aspects of a prototype. However, they can only be used to measure aspects that were predicted on beforehand, and are thus biased by the imagination of author. As a complement, open questions were also frequently used. Open questions take much more time to fill in, but they are also less suggestive. Unlike Likert scale questions, open questions can sometimes reveal unexpected UX aspects. This is important as human behavior is ultimately unpredictable, and one cannot anticipate all potential actions and behaviors in an experiment.

During experiments, observation and video recording were also frequently used in addition to questionnaires. These tools have the benefit of capturing the process, and not just the end result. Additionally, these tools require no user input, and consequently do not hamper or interrupt the user's process. However, extensive amounts of data are captured, and analysis can quickly become unwieldy and time-intensive. Still, they can reveal aspects of interaction that participants themselves were not consciously aware of.

Finally, we use certain techniques to encourage participants to vocalize their thoughts. When participants get "in the zone" (i.e. enter a state of flow), they tend to become silent, totally engrossed by the task at hand. The first technique is the think-aloud protocol (Als et al., 2005; Fonteyn et al., 1993; Kesteren et al., 2003). Simply put, participants are asked to vocalize their actions and thought patterns throughout a particular test or interaction. By doing so, they afford researchers an insight into their thought process, which can help to identify aspects of interaction that would not otherwise be clear on video. At the start of an experiment, participants may need an occasional reminder, though eventually they get used to talking in a stream-of-consciousness type mode. The second technique is called co-discovery (Kemp and Gelderen, 1996; Kesteren et al., 2003). Here, the idea is to perform experiments with small groups of participants, ideally in pairs. Because participants are asked to collaborate, they will naturally vocalize their actions and motivations toward one another. In contrast with the think-aloud method, vocalization in co-discovery is perceived as more intuitive and natural.

As a final tool, we also employed follow-up interviews in some experiments. This technique affords an opportunity to explore interesting results from a questionnaire in a deeper manner. However, interviews can be very time consuming and take some time to prepare. Consequently, in experiments where follow-up interviews were used, key participants were selected. By nature, follow-interviews happen a certain amount of time after an experiment, which can introduce problems of their own. Human memory is inherently flawed, as there is a difference between experiencing and reflecting upon an experience. This idea is mirrored by Kahneman (2013, chap. 35) in the model of the two selves: the experiencing self and the remembering self. Kahneman summarizes the concept as follows: "Odd as it may seem, I am my remembering self, and the experiencing self, who does my living, is like a stranger to me".

Table 2.3 summarizes the measurement instruments we used throughout this work. The numbering of this table will be used throughout the other sections of this work to refer to specific measurement instruments. Based on our experiences, table 2.3 also includes a

subjective appraisal of the time it takes for participants to complete the survey, as well as the complexity of the tool.

Measurement tool		Type	Speed	Ease of use	
$M_1$	System Usability Scale	quantitative	standard	short	medium
$M_2$	AttrakDiff	quantitative	standard	medium	hard
$M_3$	Pick-A-Mood	quantitative	standard	short	easy
$M_4$	Likert scale questions	quantitative	custom	short	easy
$M_5$	UX curve	qualitative	standard	medium	hard
$M_6$	Open questions	qualitative	custom	long	hard
$M_7$	Interview	qualitative	custom	long	medium
$M_8$	Video capture	qualitative	custom	long	easy

**Table 2.3** Summary of UX measurement tools

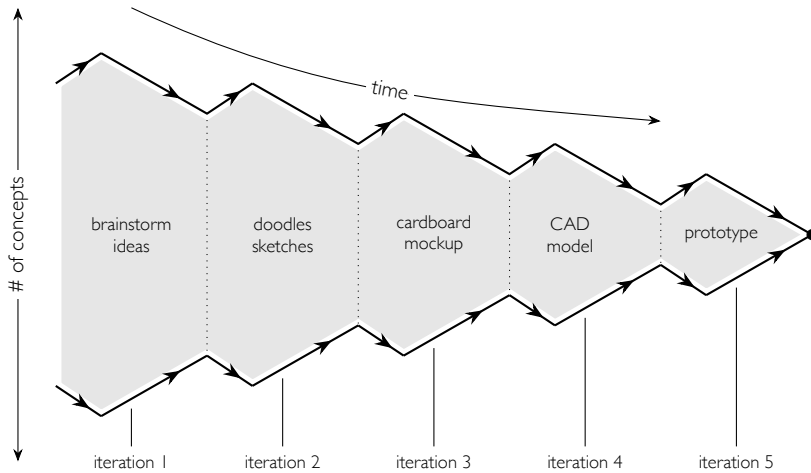
## 2.2 ITERATIVE PROTOTYPING

Designing for user experience is difficult because user experience is an emergent phenomenon that arises from the total sum of all aspects of a product. It is impossible to partition the design and optimize each of its constituent parts for user experience. As argued in the previous sections, a holistic approach must be used; a product’s UX cannot be fully understood as the sum of its parts. Within the design professions, the accepted method to tackle the challenges posed by user experience design is to use an iterative design process.

The ISO model for human-centered design (ISO, 2010), discussed earlier in this chapter, approximates the design approach of this project on a macro-level. On a zoomed-in level, our approach is best represented by Pugh’s model of controlled convergence (1991). Pugh’s model, and by extension Buxton’s adapted version (2007), approximates the design process as a series of divergent and convergent phases. During divergent phases, new design solutions are synthesized, broadening the solution space. During convergent phases, the best solutions are selected and the rest are discarded. After each divergent-convergent pair the solution space narrows, and the process is repeated until an appropriate solution is found. Consequently, each pair of divergent/convergent phases can be labeled as a design iteration.

Figure 2.11 illustrates this concept with a figurative example. It illustrates how the prototyping chain generally starts with low-effort, low-fidelity techniques and gradually transitions to more time-consuming techniques that yield higher-fidelity prototypes. For instance, a design process can start with brainstorming and loose words, then transition to sketches, cardboard mockups, CAD models, and finally a series of 3D-printed or lasercut prototypes. Each iteration expands and converges the solution space, moving the design forward.





**Fig. 2.11** Example illustrating Pugh's model of controlled convergence. Adapted from Buxton (2007) and Pugh (1991).

It is important to have many quick iterations because it allows you to identify problems before too much work is invested into an idea. The iterative process can be repeated infinitely, though the process experiences diminishing returns and the solution space approaches the ideal solution asymptotically. In practice, a product version is released when the proposed solution is deemed good enough to meet or exceed most of the user's requirements and wishes.

As argued earlier, iterative prototyping is essential for user-centered design, and digital manufacturing techniques are well suited to support this process. With the Opsoro platform, we wish to apply a software release cycle model in a hardware product. Models such as Agile software development have adopted the mantra of “release early, release often”, eschewing the traditional model of monolithic software released (Beck et al., 2001). Because digital manufacturing technologies are not tied up in large capital investments such as molds and tooling, using a continuous release cycle in hardware becomes a possibility.

## 2.2.1 DIGITAL MANUFACTURING TECHNIQUES

Digital manufacturing technologies, such as 3D printing, are fabrication technologies that can be used to make physical objects directly from digital files. This is in contrast with conventional production techniques, where there are intermediary steps between a design and a physical part. For instance, in manual fabrication, a person will use tools to shape material in the form of a design, though this necessitates skill and artisanship. Mass production technologies, on the other hand, rely on expensive tooling and molds to make parts through an automated process.

As the name implies, digital manufacturing techniques are computer-controlled, mini-

mizing user involvement and skill requirement. These technologies have enjoyed a lot of attention recently due to phenomena such as the maker movement and the open source hardware movement (see section 1.1). However, many of these techniques have been available in industry for a long time. Especially within research and development environments, rapid prototyping techniques are often used to accelerate the development of a new product. The technology is not new, the first CNC milling machines were created in the 1950s, and the first 3D printers appeared in the 1980s (N. Gershenfeld, 2012).

What is new, however, is that these technologies have started to become accessible outside of large R&D departments and research institutes. Various trends work together to make digital fabrication available to the masses:

- Low cost, open source digital fabrication machines, such as the RepRap (Jones et al., 2011), have begun to appear.
- Local organizations, such as FabLabs, makerspaces, and Techshops, offer access to digital manufacturing infrastructure to the general public.
- Rapid prototyping techniques are now also offered through online services. These companies will produce custom parts and send them to you for a fee. Examples include Shapeways, i.materialise, Sseed Fusion PCB.
- As technology matures, it becomes more affordable and better performing. This is especially true for electronics and computer hardware.

Digital manufacturing technology has a number of advantages that place them in a unique position with respect to conventional manufacturing paradigms. First of all, these machines do not require large part-specific capital investments, such as molds or tooling. Using digital manufacturing to make one-off parts costs the same as manufacturing parts in a batch, opening up opportunities for agile product development and low-volume manufacturing. Secondly, the manufacturing process is driven by digital files. These files can easily be shared and archived on the internet, greatly facilitating collaboration and sharing through online platforms such as GitHub and Thingiverse. Finally, an intrinsic property of digital fabrication is that design complexity is (nearly) free: printing a complex object takes the same time as printing a simple object of the same volume. This property can be exploited to incorporate extra functionality in the geometry of a part.

The term “digital manufacturing” covers a very broad spectrum of fabrication techniques, and not all techniques are appropriate for DIY and maker culture. In broad strokes, the working principle behind the different techniques is very similar. Most digital fabrication machines consist of a toolhead attached to a motion platform. The toolhead can either remove material (subtractive manufacturing) or add material (additive manufacturing, also called 3D printing). This toolhead is attached to a motion platform, turning it into a CNC machine. Most machines have a two-axis or three-axis motion platform, though machines with more axes of motion are also not unheard of.

Table 2.4 details the properties of several digital manufacturing techniques. This table is not meant to be an exhaustive overview of all techniques, it merely serves to illustrate the broad landscape of digital manufacturing. The data should also be taken with a grain of salt; there are large discrepancies even between machines that use the same working principle, and the properties are merely indicative of typical machines.

Within the spectrum of digital fabrication technologies, we focus on laser cutting and low-end FDM 3D printing (i.e. RepRap (Jones et al., 2011) and derivatives). These two technologies are commonly available in FabLabs and through online services. More importantly, these machines can be operated by novice users, which in contrast with other techniques, such as CNC milling, where more specialized training is needed. Low-cost 3D printing has enjoyed much attention in the past five years.

Some have even predicted that low-cost 3D printers could very well be the next big trend in home robotics (Guizzo and Deyle, 2012; Lipson and Kurman, 2013). Still, Anderson (2012) remarks that while 3D printing enjoys most of the media attention, laser cutters are the true workhorse of the maker movement. Both laser cutting and low-end FDM 3D printing techniques are used extensively throughout this work. These two techniques are complementary in nature: 3D printers are well suited for producing small, complex, three-dimensional parts, whereas laser cutters are fast and work well to produce larger, stronger parts than 3D printed parts. However, only flat parts can be made using laser-cutting. In our designs, the majority of custom parts are produced using laser cutting, supplemented with 3D printed parts for complex mechanisms and structures.

The design decision to use digital fabrication has implications for a number of the project goals. With digital fabrication techniques, the information required to produce a part is contained within a digital file. Computer-controlled machines use this data to produce physical parts, requiring little skill or artisanship from the operator. This last point is important with respect to reproducibility: it facilitates online sharing, it lowers the barrier to making copies (cfr. open source software, section 1.1.1) and offers a higher degree of repeatability. This directly impacts the openness of the system ( $G_1$ ), as well as the opportunities for online community building ( $G_{10}$ ). The choice also influences the cost of the designs ( $G_9$ ), as the techniques allow anyone to create high-fidelity robot components at reasonably low cost and in very small batch sizes. Finally, the techniques allow for a high degree of design complexity, affording designers the chance to embed extra functionality in the geometry of a part. This aspect can be leveraged to make the designs easier to reproduce ( $G_1$ ) and easier to build ( $G_2$ ). The next section will detail how design complexity can be exploited to achieve these goals.

## 2.2.2 DESIGN STRATEGIES

The scial robot Ono and the Opsoro platform have gone through multiple design iterations, from which we got different insight on the success and failures for non-experts to design and build robots. Throughout the iterations, our design strategies have been adjusted accordingly. The maker/hacker-centric approach lends itself well for quick design

Additive manufacturing			Subtractive manufacturing				
Fused deposition modeling		Stereolithography	CO2 laser	Low-end CNC mill	High-end CNC mill	PCB Manufacture	Water jet Cutting
Part size	10 - 100 mm	10 - 100 mm	10 - 1000 mm	50 - 1000 mm	50 - 250 mm	10 - 100 mm	50 - 2000 mm
Geometry	3D	3D	2D	3D	3D	2D	2D
Precision	low	low-medium	medium	medium	very high	high	high
Materials	thermoplastics	resins	plastics, woods	plastics, woods, composites	plastics, metals	composites	metals, plastics, composites, ceranics
Design skill	medium	medium	low	low-medium	high	medium-high	low
Production skill	low	low	low	medium	high	n/a	medium
Speed	slow	slow	fast	medium	medium	very slow	fast
Part cost	low	medium-high	low-medium	low-medium	high	low	high
Machine cost	low	low-medium	high	medium	high	n/a	very high

Table 2.4 Examples of digital fabrication techniques

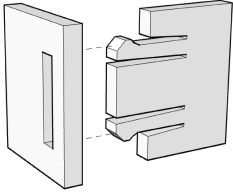
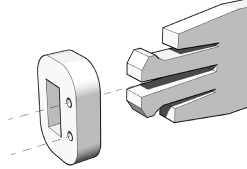
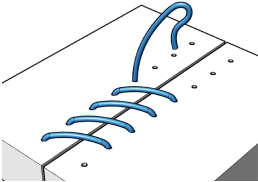
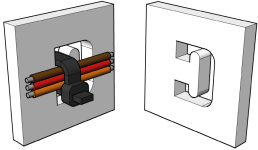
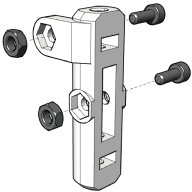
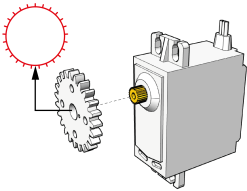
iterations due to the use of digital manufacturing techniques. A summary of this approach is given in this section. In our designs, we (1) use standardized components wherever appropriate and (2) manufacture all custom components using digital fabrication techniques. These two constraints serve to improve the reproducibility of the designs. We rely on digital manufacturing to produce the components for our designs, though other potential solutions are to create designs that can be made using hand tools or designs that use only off-the-shelf components.

The first point is self evident: using standard components is often cheaper and faster than creating a custom part for the same purpose. The second point deserves some elaboration. With digital fabrication, computer-controlled machines use digital data to quickly and accurately produce parts, requiring only limited human involvement in the process. This facilitates online sharing, lowers the barrier to copying, and offers a high degree of repeatability. Another intrinsic property of digital fabrication is that design complexity is (nearly) free, as explained earlier. This property can be exploited to incorporate extra functionality in the geometry of a part.

Within digital fabrication technologies, we focus on laser cutting and low-end FDM 3D printing (i.e. RepRap (Jones et al., 2011) and derivatives) as these two technologies are commonly available through FabLabs or online services. By taking advantage of digital manufacturing techniques we can incorporate extra functionality in our custom parts. This can lead to a reduced part count, simpler assembly, improved cable management, etc. Table 2.5 shows an overview of connections made possible through clever manipulations of custom component geometry.

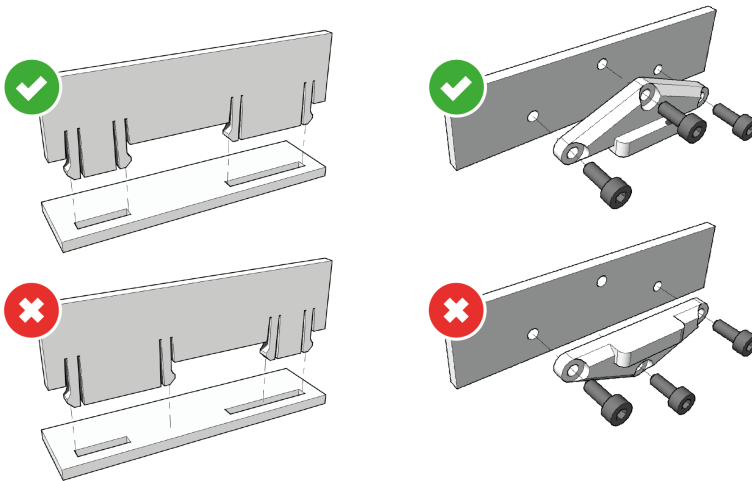
Wherever possible, assembly information is embedded into the part geometry. Multiple methods can be used for this. To begin with, all laser-cut and 3D printed parts are fitted with engraved annotations, indicating part numbers and orientation. This is useful to distinguish similar parts and helps when referring to a part in written documentation. Keeping track of different part of the same design, different versions of the same part, or even the same parts within alternative designs is an interesting problem, and will certainly become more relevant as the open source hardware community grows. As of yet, we manually add the part numbers during the CAD design process. However, we envision opportunities to automate these steps. Technically speaking, automatically adding part numbers is a relatively easy process, though these annotations can have a tremendous effect for the user. Sequential version numbering (e.g. *v1*, *v2*, *v3*) is not always possible within open source hardware designs, so alternative version numbering schemes should be considered. One possibility is to use timestamps, another is to use the commit hashes from a version control system.

Many parts are purposefully made asymmetric, so that they can only be assembled in one way. Figure 2.12 shows two examples of this. In the left-hand example, correct orientation is enforced by giving the snap connectors different widths. In the right-hand example, this is achieved by moving the middle hole out of center. Laser-cut parts are made from ABS sheet material with one textured and one smooth side. Though not a deliberate choice, the texture makes it very easy to distinguish between mirror parts. This is especially useful because laser-cut parts always have at least one plane of symmetry, parallel to the plastic

Figure	Description
	<p><b>1. Snap connector</b> (ABS – ABS)</p> <p>A reversible cantilever snap is used extensively to make 90° T-shaped connections between two laser-cut parts. Used for connecting the different parts of the frame together and for connecting modules to the frame.</p>
	<p><b>2. Textile snap</b> (ABS – Textile)</p> <p>A variant on connection 1 is used to attach the outer textile to the modules. A small laser-cut receiving part is sewn directly to the textile. This receiving part mates with its counterpart in the modules.</p>
	<p><b>3. Stitch pattern</b> (Foam – Foam)</p> <p>The foam padding layer of Ono is made from flat, laser-cut foam parts that are sewn together to form a three-dimensional shell. This 2D pattern is generated by flattening the 3D shape in software. The border of the foam parts is punctured by the laser at 1cm intervals, facilitating the manual sewing process.</p>
	<p><b>4. Zip tie “dog bone”</b> (ABS – Cable)</p> <p>Feature to attach wiring to the frame using a zip tie. Previously, two parallel slots were used but this proved troublesome as zip ties needed to be inserted before assembly. Because of the “dog bone” shape, zip ties can be attached post-assembly.</p>
	<p><b>5. Nut trap</b> (3D print – Fastener)</p> <p>Hexagonal pockets are used in 3D printed parts to connect to fasteners. A hex nut is pressed into the pocket, after which something else can be attached to the printed part using a screw. This method is much more reliable than cutting threads directly into the printed part.</p>
	<p><b>6. Servo spline</b> (ABS – Servo)</p> <p>Hobby servos use a splined output shaft with triangular teeth to transfer torque to the output lever. While the dimensions of servos are standardized, those of the horns that come with servos are not. We solve this by cutting a radial pattern of lines in the circumference of a circular hole. This melts the plastic in such a way that mates with triangular teeth of the spline. Cutting the triangular teeth directly does not work, as the geometry is too small.</p>

**Table 2.5** Examples of using design complexity of digital manufacturing techniques

sheet. Laser-cut parts of the same sub-assembly are also left connected to each other via small bridges, similar to a sprue tree of a model kit.



**Fig. 2.12** Using asymmetry to improve the assembly process.

Admittedly, there are limitations to this approach. The embedded information is not sufficient to document the assembly process completely. The main documentation is provided through a wiki, including photos, written instructions, and 3D models. During the workshops we organized, we found that one of the best ways to explain the assembly process is to bring an assembled example that participants can copy. Nevertheless, this is only possible in specific situations, and does not help to propagate the design via internet.

## 2.3 CONCLUSION

This chapter has described the methodology used throughout this work. Because of the complexities of platform-based products, we argue that a human-centered design process needs to be applied, with an emphasis on usability and user experience aspects. Human-centered design requires a holistic approach, as different product aspects cannot be extracted and individually optimized. For this reason, we employ an iterative prototyping process through the use of digital manufacturing technologies such as laser-cutting and 3D printing. These technologies afford the designer a large degree of design freedom, which can be used to facilitate the assembly process ( $G_2$ ). The digital manufacturing techniques are commonly available in FabLabs and through online services, facilitating reproduction and modification of the design ( $G_1$ ) and leading to lower part costs ( $G_9$ ). The next chapter will detail the iterative design process, including a description of the design process and the experiments we have performed.





## Chapter 3

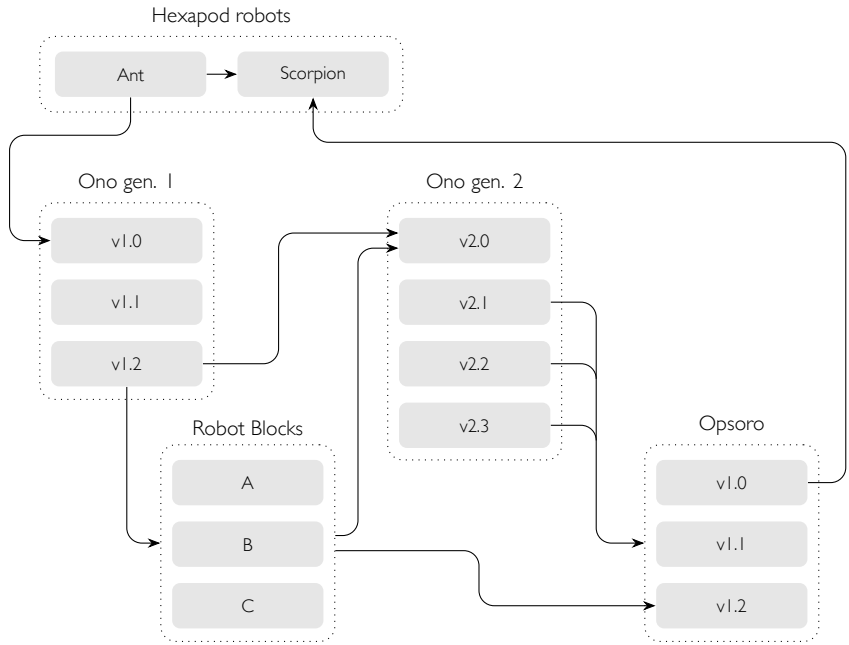
# DESIGN ITERATIONS

Ono and the Opsoro platform have gone through multiple design iterations, from which we gained different insight into the success and failures of different approaches. The design iterations explore the different aspects of the *design*, *build*, and *use* phases.

As explained by the previous chapter, the complex nature of platform-based products requires us to use iterative design in order to fully comprehend and address the needs of the different stakeholders. The challenge is to find a good balance between the conflicting goals of different stakeholders, as well as the technical constraints of the system. Our maker/hacker-centric approach lends itself well for quick design iterations, especially due to the use of digital manufacturing techniques. A summary of the major outcomes of this iterative approach is given in this section.

Figure 3.1 shows a schematic overview of the design iterations of the Opsoro platform. The main design path is represented by the two generations of the social robot Ono combined with the Opsoro platform. The figure also depicts two side-tracks: the hexapod robots and the Robot Blocks toolkit. These tangents are not focused on social robotics per se. Nevertheless, they served an important role in inspiring and informing the design of the platform. The figure illustrates the non-linear nature of our design process; the reasoning behind this is argued in chapter 2. The remainder of this chapter is structured as follows:

- **SECTION 3.1** describes the design of walking hexapod robots. The first version was created at the start of this project, and many of its construction techniques were used in the design of the initial version of the social robot Ono. The second hexapod robot was created much more recently, and incorporates improvements from the Ono and Opsoro design iterations.
- **SECTION 3.2** details the design and use of a DIY construction system for making small-scale mobile robots, intended for use by secondary school students.
- **SECTIONS 3.3 AND 3.4** explain the global design decisions behind the social



**Fig. 3.1** Schematic overview of the design iterations.

robot Ono, as well as the test results that informed these decisions. The description of Ono is divided into two separate sections, representing the complete redesign of the robot that took place midway through the project.

- **SECTION 3.5** details the shift in research focus from a single social robot design to a platform for the design of social robots. The toolkit, which is based on the Ono robot, has gone through multiple iterations, and each iteration was tested in a robot design context.

### 3.1 HEXAPOD ROBOTS

#### 3.1.1 STIGMERGIC ANT

The Stigmergic Ant came about in 2012 as a demonstrator for the different types of prototyping technologies offered at the department’s industry services division, including CNC milling, thermoforming, 3D printing, laser cutting, and electronics prototyping. The key challenge of this project was to integrate these different technologies into one showcase that would demonstrate the benefits of each technology in an interesting and attractive format. The demonstrator is designed primarily for use in exhibitions and trade fairs.

The demonstrator borrows the concept of “stigmergy” as a metaphor for the department’s approach to industrial design. The term “stigmergy” was coined by Grassé (1959) as a name for the process used by eusocial insects, such as ants, to coordinate their actions. The term is composed of the Greek words *stigma* (mark, sign) and *ergon* (work, action), denoting that agents in social insect colonies leave marks in their environment as a way to communicate future actions.

The concept of stigmergy is transposed onto the design process. Here, physical prototypes are the marks that are left in the environment, and the process of prototyping is seen as a rich and meaningful mode of communication between the stakeholders of a design process. In our industrial design department, we implement this philosophy by emphasizing the importance of making physical prototypes early and often, as part of an iterative communication process between stakeholders.

The Stigmergic Ant is conceived as the literal embodiment of this figurative metaphor for industrial design, as a sort of mascot. The ant is implemented as a 60 cm long hexapod robot with three degrees of freedom in each leg. As a tongue-in-cheek interpretation of the stigmergy paradigm, the Stigmergic Ant leaves traces in its environment in the form of a candy trail.



**Fig. 3.2** Final prototype of the Stigmergic Ant

### 3.1.1.1 EMBODIMENT DESIGN<sup>1</sup>

The design of the Stigmergic Ant was influenced in a large part by the mechanical design of the leg. The leg was designed first because (1) it is a large subassembly that is used six times, (2) the components of the leg represent the majority of the total cost of the robot and (3) most design constraints, such as overall size, weight, range of motion, and power supply, are a direct consequence of the leg.

We decided to use three actuators for each leg. While hexapods with two DOF legs are possible, the extra DOF allows for full six-axis control of the body of the robot and also allows for more efficient and more natural looking locomotion. Standard-sized Radio-Controlled (RC) servos are used throughout the design, though the design necessitated the use of high-torque metal-gear servos. Each servo generates up to 15 kgf · cm of torque. The relatively high torque output is needed because during certain phases of the gait cycle, the entire weight of the robot needs to be supported by only three servos. The servos cost circa 20 € each, totaling up to 360 € for all legs. This represented a significant fraction of the budget for this project.

The size of the Stigmergic Ant was determined by the legs to keep the overall proportions in line with those of a biological ant. The leg length is largely defined by the servo actuators, which are positioned at the joints of the legs. The servos need to be spaced far enough so as not to collide with one another, yet close enough so that the mechanical advantage of the leg segments does not cause the servo to exceed its torque rating.

In addition to the 18 DOFs of the legs, the ant also has actuated mandibles, an actuated neck and a dispensing mechanism in the abdomen for a total of three extra DOFs, bringing the total to 21 servos. The two extra DOFs allow for more expressiveness and give the ant some life-like behavior. For instance, while turning, the ant will also turn its head in the direction it is heading. The mechanism in the abdomen of the robot is used to dispense a candy trail as a form of stigmergy. The mechanism uses a rotating disk to dispense M&M's from an internal reservoir. Each completed back and forth sweep drops one M&M, allowing for precise control.

Most of the custom parts for this project were made using laser cutting. Nearly all parts for the frame and the legs of the ant were cut from 6 mm Medium-Density Fibreboard (MDF), which was painted black afterwards. The laser-cut parts are interconnected using slot-and-tab joints, which are reinforced using wood glue. Servos are attached to the frame using machine screws. The thorax is composed of a double-plate design for increased torsional stiffness, as it is subjected to large loads at the base of each leg.

To improve the robot's load-bearing capacity, the base of each leg features double bearing design with a plain bearing at the opposite end of the servo's output shaft. The tip of each leg is rubberized in order to reduce slippage, seeing that friction plays a large role in the efficiency of locomotion. Finally, to accommodate the large amounts of wiring in this

<sup>1</sup> Significant contributions to the design and construction of the Stigmergic Ant were made by Ivo Six and Laurenz Tack.

robot, mounting holes for zip-ties were integrated in each segment of each leg. This serves to reduce strain on the servo leads, as well as eliminate the possibility of a cable catching and breaking.

The hollow abdomen of the Stigmergic Ant is composed of a laser-cut rib structure on the outside, with a two-part plastic shell on the inside to serve as a candy reservoir. The shell was modeled as a Non-Uniform Rational B-Splines (NURBS) surface in CAD. Using this surface, a positive mold was milled in Polyurethane (PU) foam, which was subsequently used to thermoform the two parts out of ABS plastic. The seam between the two halves is hidden by the outer ribs.

The electronics of the ant robot are Arduino-based, with an Arduino mega handling communication, gait pattern generation and inverse kinematics. A separate SSC-32 servo controller<sup>2</sup> handles the Pulse-Width Modulation (PWM) waveform generation to control the 21 RC servo. The SSC-32 itself is controlled by the Arduino Mega over UART.

Initially, the ant was powered through a tether cable, which connected the robot to an external high-current 5 V power source. The tether was also used for communication with the computer-based GUI. This setup was eventually upgraded to eliminate the tether. The external power supply was replaced by an integrated 6 V NiMH battery pack, and the communication cable was replaced by an XBee wireless link. Besides the obvious benefits, this upgrade also improved the stance and speed of the robot, as removing the thick-gauge cable made the robot noticeably less tail-heavy.

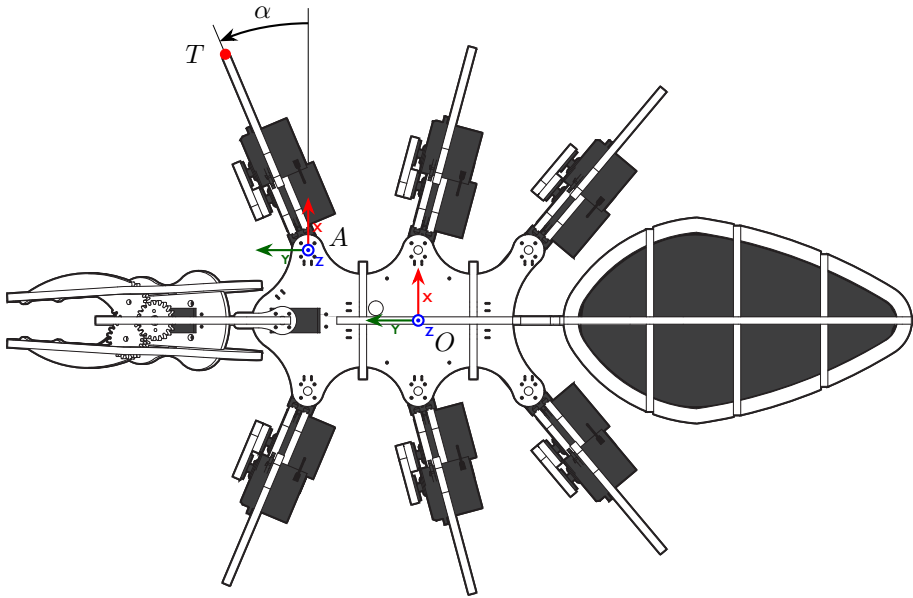
### 3.1.2 LOCOMOTION ALGORITHM

Each leg of the hexapod has a horizontal hinge joint near the base of the body, followed by two vertical hinge joints (fig. 3.4;  $\alpha$ ,  $\beta$ , and  $\gamma$  respectively). This configuration enables three-axis positional control of the leg tip. Gaits and body position/orientation control are made possible by coordinated movement of the leg tip positions of all six legs. In essence, the control scheme relies upon looking at motion of the robot as a transformation of the world relative to a *stationary* robot body, and transforming leg tip positions to coincide with the *moving* world.

To accomplish this, the software needs a method to determine a leg's joint angles for any given set of leg tip coordinates. Algorithms that perform such calculations are referred to as Inverse Kinematics (IK) algorithms. More complicated kinematic chains, like those found in a six-axis industrial robot arm, rely upon iterative approximation IK algorithms to find arm solutions. To avoid this, the mechanical design of the ant's legs was modified so that the IK problem would have an exact analytical solution. This resulted in a dramatic simplification of the IK algorithm. Consequently, the Stigmergic Ant needs significantly less computational power. The algorithm imposes one important requirement on the mechanical design of the leg: the leg tip needs to coincide with the plane that is perpendicular

<sup>2</sup>Lynxmotion - SSC-32 Servo Controller –  
<http://www.lynxmotion.com/p-395-ssc-32-servo-controller.aspx>

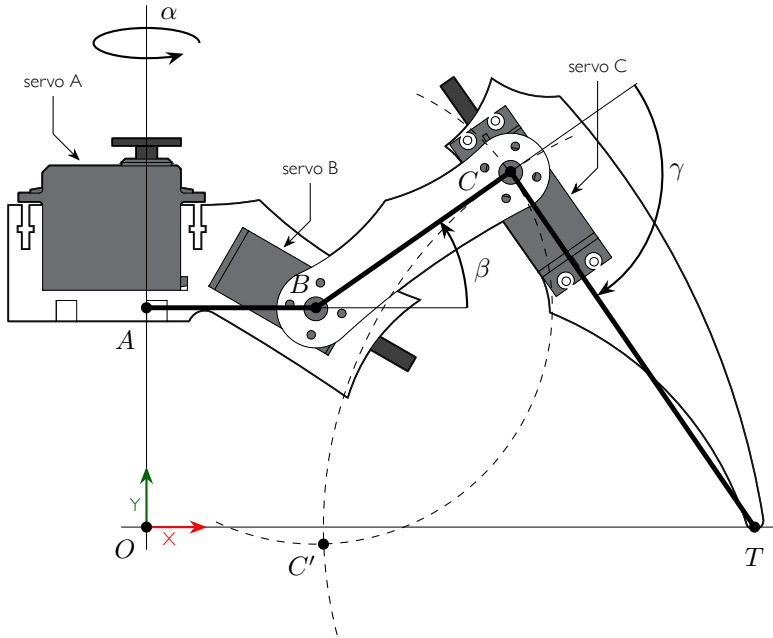
to the axes of rotation of servos B & C, and contains servo A's axis of rotation (see fig. 3.4 for servo naming). As a result, the three segments of the kinematic chain of a leg become coplanar, allowing for a simple, analytical solution using trigonometric functions.



**Fig. 3.3** Top view showing tip position at  $T$ , body coordinate system at  $O$ , and leg coordinate system at  $A$

The complete IK algorithm can be summarized in the following steps:

1. The IK algorithm is fed six 3D points in the ant's body coordinate system. These points represent the desired leg tip positions. The origin of the body coordinate system coincides with the center of the robot's body, as shown in fig. 3.3. Leg tip positions are determined by a higher-level alternating tripod gait algorithm.
2. For each leg, the tip position  $T$  is transformed from the body coordinate system at point  $O$  to the corresponding leg coordinate system, which has its origin at point  $A$ ; the center of rotation of servo  $A$ .
3. The angle of rotation of servo  $A$ ,  $\alpha$ , can be determined easily. It is the angle between the leg coordinate system  $x$ -axis and the projection of segment  $\overline{AT}$  onto the  $XY$  plane (fig. 3.3).
4. The lengths of segments  $\overline{AB}$ ,  $\overline{BC}$ , and  $\overline{CT}$  are known, they are determined by the geometry of the leg. These lengths are used to construct two circles, the first with radius  $\|BC\|$  around point  $B$ , the second with radius  $\|CT\|$  around point  $T$  (fig. 3.4).

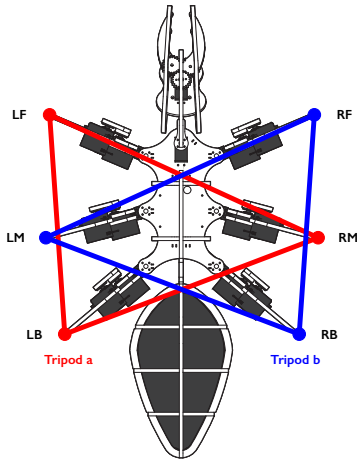


**Fig. 3.4** Kinematic chain of the leg

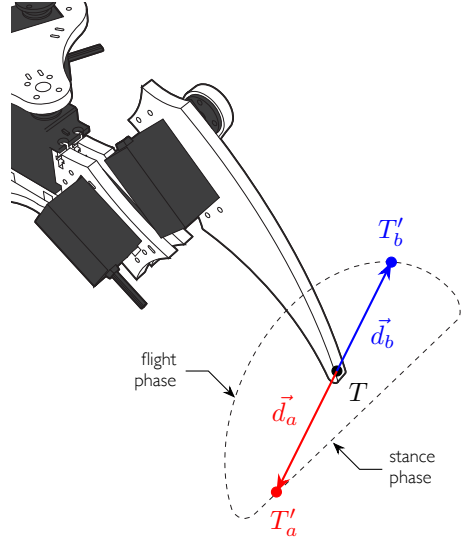
5. Next, the position of point  $C$ , the knee, is determined. It follows from the leg geometry that the knee point should be situated at the intersection of the two circles. For every valid (reachable) tip position, there are two solutions, points  $C$  and  $C'$ . The solution with the smallest  $z$ -coordinate,  $C'$ , is discarded.  $C'$  is situated below ground level and cannot be used.
6. Once the position of point  $C$  is determined, angles  $\beta$  and  $\gamma$  can be calculated using inverse trigonometric functions.
7. Finally, angles  $\alpha$ ,  $\beta$ , and  $\gamma$  are mapped from  $[-\frac{\pi}{2}, +\frac{\pi}{2}]$  to  $[500 \mu\text{s}, 2500 \mu\text{s}]$  and a calibration offset is applied. The resulting pulse width values are sent to the servo controller.

The locomotion of the Stigmergic Ant is generated using a simple alternating tripod gait algorithm. In this gait, legs are grouped into two sets of three, the eponymous tripods (fig. 3.5). The first set is composed of the left front leg (LF), the right middle leg (RM), and the left back leg (LB). The second set contains the opposing legs. The legs of each set move simultaneously, and there are always three legs in contact with the ground.

This gait is commonly employed by insects while moving slowly, and was first described by Wilson (1966). The alternating tripod gait is relatively well understood and is commonly implemented in hexapod robots using a variety of different techniques (Altendorfer et al.,



**Fig. 3.5** Tripods of the alternating tripod gait



**Fig. 3.6** Effect of the displacement vectors on leg tip positions

2001; Beer et al., 1997; Birkmeyer et al., 2009; Wang et al., 2010). The characteristics of this gait are that it is statically stable and relatively simple, though slower than other gaits.

The Stigmeric Ant implements this alternating tripod algorithm by superimposing a displacement vector upon the neutral position of each leg tip (fig. 3.6). The displacement vectors  $\vec{d}$  are precalculated and describe a semi-circular path with a linear segment for the stance (push off) phase and an arc segment for the flight (return) phase of the gait. Displacement vector coordinates are stored in a circular array.

The displacement vectors of the two tripods,  $\vec{d}_a$  and  $\vec{d}_b$ , are in antiphase, so that whenever one tripod is in stance phase, the other is in flight phase. At every time step, the indices of the displacement vector array are incremented, resulting in displacement vector  $\vec{d}_a$  for the legs of tripod *a* and displacement vector  $\vec{d}_b$  for the legs of tripod *b*. For each of the robot's legs, a new tip position  $T'$  is found by superimposing the displacement vector  $\vec{d}$  upon the leg's neutral position  $T$ , so that  $T' = T + \vec{d}$ . For the legs of tripod *a*, this becomes  $T'_a = T + \vec{d}_a$ . For tripod *b*, the result is  $T'_b = T + \vec{d}_b$ .

The steps described above are used to implement the ant's forward locomotion. Turning is built upon the same principle, but the process is slightly more complicated. Here, the x-component of the displacement vector is used to drive rotation of the leg tip about the robot's body origin  $O$ , and the z-component is used to translate the leg tip vertically. As a result, the two tripods rotate in opposite directions around the body origin, causing the robot to turn in place. A more generalized approach could incorporate rotation and translation in the same transformation matrix, enabling simultaneous changes of position



and orientation, though this has not yet been implemented.

### 3.1.3 SCORPION<sup>3</sup>

In early 2016, the Stigmergic Ant was disassembled and components were reused to create a second-generation hexapod robot, this time inspired by the design of a scorpion (Terry et al., 2016). Improvements of the scorpion over the Stigmergic Ant fall into two categories:

1. The high-level software of the robot was changed to have a large degree of autonomous, animal-like behavior. This is in contrast with the original Stigmergic Ant, which functioned much more as a remotely operated robot. This new, semi-autonomous mode of operation requires much less space to deploy, making it more useful for expositions and trade shows, where space is at a premium.
2. The mechanical design of the robot is greatly improved, leading to reduced weight, longer battery life, and overall better performance.

The interaction paradigm of the scorpion was changed with two things in mind. First of all, the robot should be usable in an area as small as a tabletop. Secondly, interactivity should be focused on very short interactions with visitors, and when no one is actively playing with the robot, the robot should display some sort of attention-grabbing behavior.

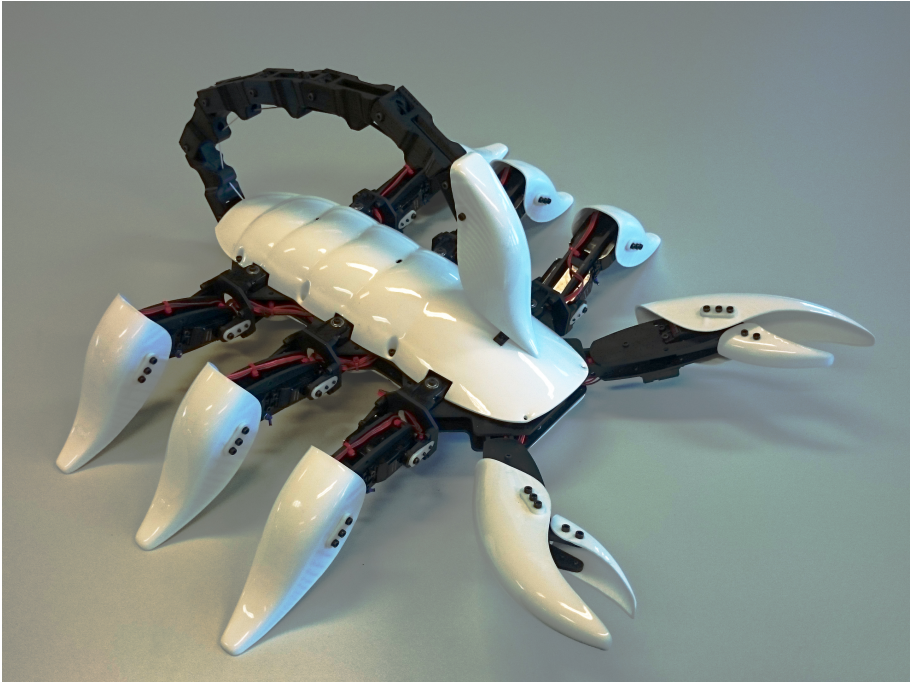
Toward this end, a number of IR sensors were integrated into the body of the scorpion. Sensors positioned on the ventral side of the robot function to detect and avoid table edges, whereas sensors on the dorsal side are used to detect a user's hand. Toward this end, the robot has 3 IR detectors along the length of its back, which detect the presence of a hand above it, as well as one IR distance sensor in front, which is used to measure the distance between the hand and the robot.

When no one is nearby, the scorpion exhibits a seeking behavior, where it will randomly turn and look for a victim. When a visitor positions their hand over the head of the scorpion, it will assume a defensive, threatening stance warning the user to back away. Should a visitor choose to ignore this warning, the scorpion will sting, marking them with a felt marker hidden within its stinger. Conversely, the robot may also be calmed down by stroking its back, in which case, it will exhibit a happy, soothing behavior.

The second group of improvements is related to the mechanical design of the hexapod. We discern three major factors that triggered the changes:

- Lessons learned from the Stigmergic Ant; especially problems that only arise after extended use of the device, such as servos overheating and burning out.

<sup>3</sup>The scorpion was created by Robbe Terry, Stephan Flamand, and Pieterjan Deconinck. My own involvement in this project was limited to an advisory role.



**Fig. 3.7** Final prototype of the scorpion

- The proliferation of digital manufacturing techniques. In the four years since the ant was designed, the number of digital manufacturing machines at the department has increased considerably. The capacity for existing techniques, such as laser cutting, has doubled or tripled, and new techniques, most notably low-cost 3D printing, have been added.
- Better knowledge of digital manufacturing. Expertise has been built up over the years, leading to faster design cycles, better use of materials and techniques, and a better understanding of part performance.

The first major change from the ant is the use of 3 mm ABS instead of 6 mm MDF. Though less rigid, this new material has a much better strength-to-weight ratio, permitting a more lightweight design. The scorpion has a similar double-plate chassis design, though the plates are held at distance by threaded spacers, improving chassis rigidity.

Two big changes were made to the design of the legs to reduce strain on the servos. First of all, idler bearings were added on the backplate of each servo (the side opposite the output shaft). This serves to eliminate any misalignment in the leg joints, which in turn reduces the required amount of torque. The idler design was copied from the Opsoro Joint module (section 4.1.1.4), integrating 3D printed parts in the design of the legs.

Secondly, gravity compensation springs were added to the B-axis of each leg. The servos

on this axis are subject to the highest loads and had a tendency to eventually fail due to excessive strain. Strong elastic cords were added to the underside of each leg in order to create a force opposing gravity.

Thermoforming was used more extensively in the scorpion, both for aesthetic parts as well as for structural components. The third segment of the legs (segment  $\overline{CT}$ ) was entirely replaced by a single thermoformed part made out of 2 mm polystyrene (PS). The doubly curved shape of this monocoque part gives it the strength it requires while being very lightweight. Use of thermoforming is repeated in the scorpion's dorsal and ventral cover plates, as well as its claws and stinger.

The end result of all these embodiment changes is a high-performance, lightweight robot. To illustrate, the Stigmergic Ant had an idle current consumption of 3 to 4 A while standing, while the scorpion only draws 1 A under similar conditions. Even more so, because of the antagonistic springs that were integrated into the legs, the robot can remain standing even when power is turned off. All in all, the result is a hexapod with an extended battery life and a greatly improved gait.

### 3.1.4 SUMMARY

The two hexapod robots presented in this section demonstrate the benefits of digital manufacturing techniques in complex, mechanically demanding applications. The rise of these techniques has significantly lowered the cost of creating custom parts for robotics ( $G_9$ ), with the added benefit that designs can be shared easily via online platforms, facilitating worldwide collaboration ( $G_{10}$ ). Still, the documentation and communication of a design remains a difficult and time-consuming challenge. This problem is exacerbated by (1) the cost of most CAD tools, and (2) the fact that CAD files cannot capture all the design information. For instance, component wiring, assembly order, and calibration routines are difficult to document in CAD.

With both hexapods, laser cutting served as the production technique of choice for the bulk of the components. While laser cutting has some design limitations – e.g. being limited to flat parts – the technique enables designers to rapidly create large, strong parts with an excellent finish. While the base techniques of the two robots are the same, the scorpion demonstrates a much better understanding of the laser cutting technique, as shown by the use of more durable materials and more complex geometry. This resulted in a much lighter and better performing robot. In addition, the scorpion uses thermoforming more extensively in the design, and also incorporates 3D printing for many small, complex parts, a technique that was not present in the Stigmergic Ant. Though the hexapods were not explicitly designed as open hardware, the tools and techniques used are well suited for open DIY reproduction ( $G_1$ ). The use of laser cutting and 3D printing techniques in a demanding scenario proved useful and inspired future designs in this work.

While not designed with this in mind, aspects of design modularity emerged in both hexapods. In essence, the leg – a subassembly that is reused six times – can be seen as a

“primitive” toolkit module. The modular design of the leg makes it easier to build and to use ( $G_2$ ). There is a direct parallel between the physical leg module, a reusable hardware block, and the leg driver class, a reusable software component. In both cases, complex behavior is abstracted away in an easy-to-use “black box”. A similar strategy also emerged in the Opsoro platform.

The design of the scorpion’s legs is based on a specialized version of the Opsoro joint module. It is interesting to note how this module was repurposed as a leg module, for which it was not originally intended. This is characteristic for the chosen components and techniques, which place an emphasis on flexibility, adaptation, fast design iterations, and an open, “hacking” design approach ( $G_1$ ).

## 3.2 ROBOT BLOCKS – TOOLKIT FOR SIMPLE EDUCATIONAL ROBOTS

Building a robot from scratch in an educational context can be a challenging prospect. While a multitude of projects simplify the electronics and software aspects of a robot, the same cannot be said for construction systems for robotics. This section presents our efforts to create a low-cost do-it-yourself construction system for small educational robots. Currently, one of the most common ways of implementing robotics in the school curriculum is through the use of commercial robot kits, such as LEGO Mindstorms. The alternative to using a commercial robot kit is building a robot from scratch. In the past, this would have meant selecting microcontroller and motor driver chips, developing a printed circuit board, CNC milling a custom chassis for the robot, and finally programming the robot in low-level languages such as assembly or C. In recent years, however, a number of projects have come about that greatly simplify this process: they provide flexible, yet user-friendly solutions to each of the sub-problems presented in robotics. A complete robotics platform can be seen as the combination of three distinct elements: (1) a set of electronics, (2) a programming environment (software), and (3) construction elements for a physical embodiment. As discussed in section 1.2.2, multiple projects exist that provide an intermediate solution between building from scratch and using a commercial kit. In doing so, they provide user-friendly tools for educational robotics that can potentially lead to a deeper learning experience. While this approach is already successful and widely used in software and electronics, common construction systems are still lacking. The choices for robot construction systems are not quite as diverse as those available for software and electronics. Consequently, the options that remain are either (1) to use one of the few purpose-built systems (e.g. MakeBlock, BitBeam), (2) to modify toy construction systems (e.g. LEGO, Meccano), or (3) to build a robot embodiment from scratch using raw materials. Often, cost and availability of parts are the main obstacles here. In our opinion, none of the approaches are analogous to what is offered in the other categories: platforms that allow novices to work with complex technologies in a user-friendly way, but that also allow advanced users to modify, extend, and hack the platform beyond its original capabilities. More research into the specifications of DIY robot construction systems is important here. In our work, we emphasize the role of user experience and user-friendliness precisely

because we think those are decisive factors in the success of an open construction system.

Our goal is to create an extendable construction system that can be used in conjunction with electronics and software to build small robots from scratch. Influenced by the apparent flaws of existing approaches, we paid special attention to two key aspects. First of all, cost is often a barrier in the implementation of robotics in education (Gonzalez-Gomez et al., 2012; Johnson, 2003; Mataric et al., 2007; Mondada et al., 2009; Riojas et al., 2012). As such, we have made a conscious effort to reduce cost without limiting functionality by repurposing standard components and by using affordable low-volume production methods. Secondly, we aim to make our system open (i.e. "hackable", suitable for DIY), meaning anyone should be able to modify and expand the system. In order to meet this requirement, we have restricted manufacturing techniques to those that can commonly be found in FabLabs (Walter-Herrmann and Büching, 2013), e.g. 3D printers and laser cutters. While mass production techniques, such as injection moulding, can produce parts at a much lower cost, they would significantly hinder the ability for anyone to customize parts due to the costs associated with tooling and moulds.

Even if there are no FabLabs in the vicinity, online services (e.g. Shapeways, Ponoko, 3D Hubs) offer complete access to digital fabrication techniques. However, creating parts at a FabLab has the added benefit of bringing students and teachers in contact with a new environment that offers a plethora of STEM teaching opportunities (Blikstein, 2013a). It should also be noted that the two aspects mentioned above are, in fact, interrelated. By designing the system so that students can manufacture their own parts at a local FabLab, costs can be greatly reduced. Additionally, the students become familiar with the manufacturing process, lowering the barrier to modify existing designs into custom components and introducing them to the DIY culture.

The following sections will detail the design and evaluation of this construction system. The project has undergone two design iterations, resulting in three distinct construction systems. The three systems were given to students and teachers to be used in a robotics contest. Subsequently, the systems were evaluated on three aspects, usability, affective appraisal, and functionality, through (1) surveys filled out by the participants of the contest and (2) through an expert evaluation. Based on this information, we have chosen one system to be used as the basis for the next design iteration, incorporating feedback from both the user surveys and the expert evaluation.

### 3.2.1 ROBOTS TO MOTIVATE STUDENTS INTO STEM

In the summer of 2013, Dwengo VZW<sup>4</sup> launched the CERobotics<sup>5</sup> project in Argentina. Within this project, which was co-funded by the Google RISE 2013 program, students and teachers were trained to build robots. The idea is that by hands-on experience, students and teachers evolve from being consumers to technology producers. In total, 86 students (aged 11-18) and 35 teachers participated. The project took place in the Salta

<sup>4</sup>A non-profit organisation that promotes science and technology. – <http://www.dwengo.org>

<sup>5</sup>CERobotics project documentary. – <http://www.youtube.com/watch?v=jP-G10rR5Ng>

province in northern Argentina. Due to the geographical location and low budgets, these students had minimal access to the latest technologies. For these students, robotics and the combination of electronics, mechanics, and programming was a completely new experience.

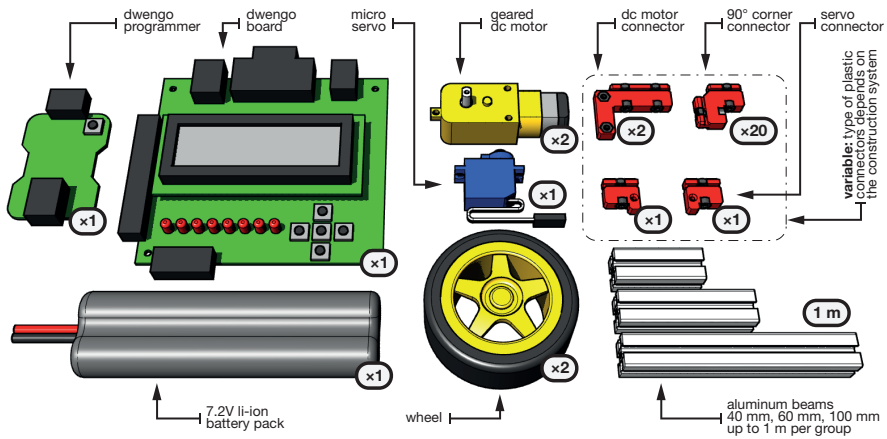


**Fig. 3.8** Playfields used during the contest. The left playfield has a light-to-dark gradient on the floor to guide robots toward the flame. The right playfield has a bright spotlight above the flame to aid the robots in navigation.

At the end of the hands-on sessions, a robot contest was organized with the challenge of designing a firefighting robot. Participants were given five days to create a robot that can autonomously navigate around obstacles towards a fire – represented by a lit candle – and extinguish it. In order to persuade students to explore different solutions, two different playfields (fig. 3.8) were provided. The first (fig. 3.8 left) has a gradient floor, so grey scale sensors can be used to determine the distance of the robot to the candle. The second (fig. 3.8 right) has a spotlight above the candle pointed towards the robot, which can be detected using IR sensors.

In total, 86 students and 35 teachers participated in the project. They were divided into teams of two to three and were given a kit with building materials for their robots. Teams consisted solely of either teachers or students; there were no mixed teams. The kit (fig. 3.9) consisted of the following items:

- 1× Dwengo Board (wyffels, Bruneel, et al., 2012; wyffels, Hermans, et al., 2010) (a microcontroller board with provisions to control 2 DC motors and 2 RC servos, along with multiple sensors).
- 1× programmer with USB cable.
- 1× battery pack.
- 2× geared DC motors with wheels.



**Fig. 3.9** The robot kit used in the contest contains a microcontroller board and programmer; a battery, 2 geared DC motors, 1 RC servo, 2 wheels, 1 m of aluminium extrusions, and a set of plastic connectors. The type of connectors (shown in red) changes between the three systems.

- 2× IR sensors.
- 1× fan and balloons (either of which can be used to extinguish the candle).
- Aluminium beams: participants were free to select pre-cut pieces of 40 mm, 60 mm, and 100 mm. A total length of 1m was provided per team. No team opted to cut beams of custom length.
- Plastic connectors, type depending on the system assigned to the team: 2 motor connectors, 1 pair of servo connectors and 20 90° corner connectors. More connectors were available, if needed. This part of the kit was the focus of the experiment.
- Miscellaneous items, such as nuts, screws and wires.

To successfully build a firefighting robot, it is important to master the basics of many STEM disciplines. Knowledge of materials, mechanisms and mechanical engineering principles are essential skills for building the physical embodiment of the robot, while the electronics require insight into electricity and physics. Finally, to program the robot, subjects such as computer science, mathematics, and algorithmic thinking are required. As such, we believe robotics contests such as this one have great educational value because they teach the basic principles of engineering, programming and electronics in a fun and engaging context (Osborne et al., 2010; Pack and Avanzato, 2004; Verner and Ahlgren, 2004; wyffels, Hermans, et al., 2010).

### 3.2.2 DESIGN OF THE BUILDING SYSTEMS

We have created our systems primarily to facilitate the design and construction of small educational robots. As such, the design decisions we have made resulted in systems that are much better suited for building small wheeled robots as opposed to flying drones, or even humanoid robots. At the start of the design process, we decided to build our construction system around standardized 15x15 mm aluminium T-slot extrusions. We based this decision on a number of factors:

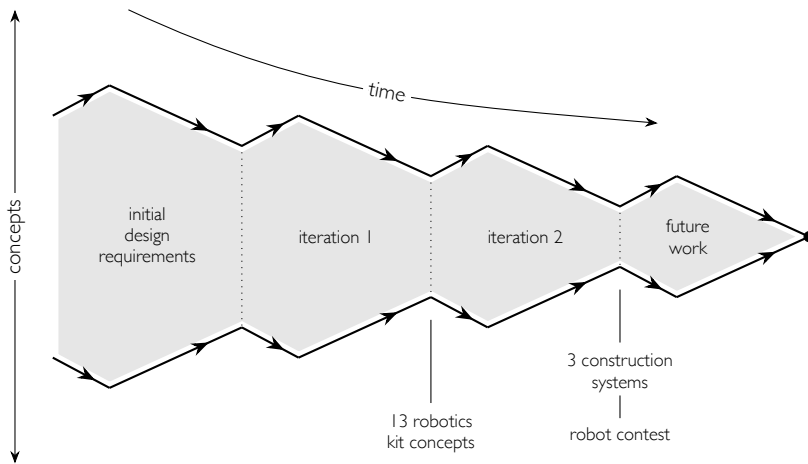
- By relying on these aluminium extrusions, the only custom components required are small connector pieces. These can be quickly and easily manufactured using a laser cutter or a 3D printer. This strategy greatly improves the machine time required to produce a kit.
- Because of the T-slots, components can be fastened at any arbitrary spot along the length of the extrusion. Systems that rely on beams with regularly spaced holes (e.g. LEGO Technic, BitBeam) do not offer this advantage.
- They are compatible with standard M3 fasteners, as opposed to larger size extrusions, which generally use proprietary nuts and bolts.
- Aluminium is stronger and more robust than common types of plastics (such as ABS or PP).
- They are inexpensive (8.60 € for a length of 2 m) and can be bought from multiple suppliers (e.g. Misumi, OpenBeam).

As mentioned earlier, one of the goals of this project is to create a system that is accessible and easy to (re)produce. As such, we have limited ourselves to tools and machines that are commonly found in FabLabs.

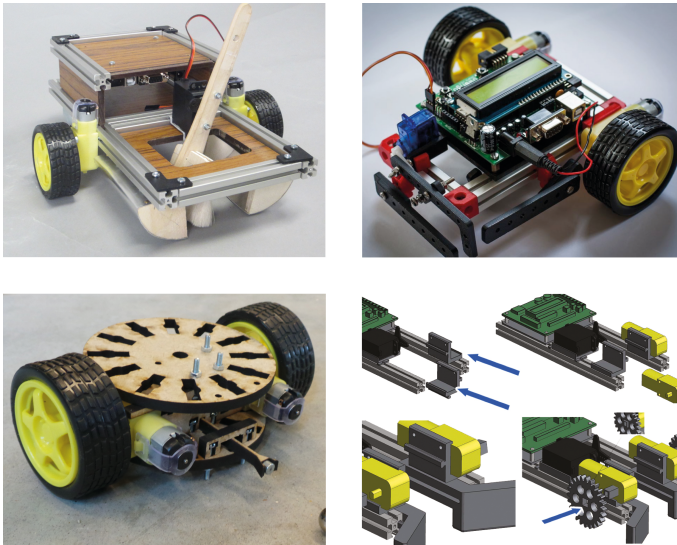
Our design approach (fig. 3.10) can be approximated by the Pugh's Design Funnel model (Pugh, 1991), with two iterations of divergent and convergent ideation. The initial design requirements can be summarized as; (1) An open DIY construction system for robot kits (2) producible with common facilities in FabLabs (e.g. hand tools, laser cutters and basic 3D printers) (3) using the standard  $15 \times 15$  aluminium T-slot extrusions. An exploratory first iteration was done in conjunction with 2nd year Industrial Design students at Ghent University. As an assignment for one of the courses, they were required to design a kit, based on the aluminium beams that could be used to build two different robots. Altogether, the students designed 13 different robot kits. Figure 3.11 shows some of the robots they created.

A number of conclusions were drawn from the students' kits. None of the connection systems they designed were suitable for a larger scale experiment. Some of the connectors showed a lack of strength, some required too much manual labour to produce, and some were simply too limited. At a certain point in their design process, the students needed





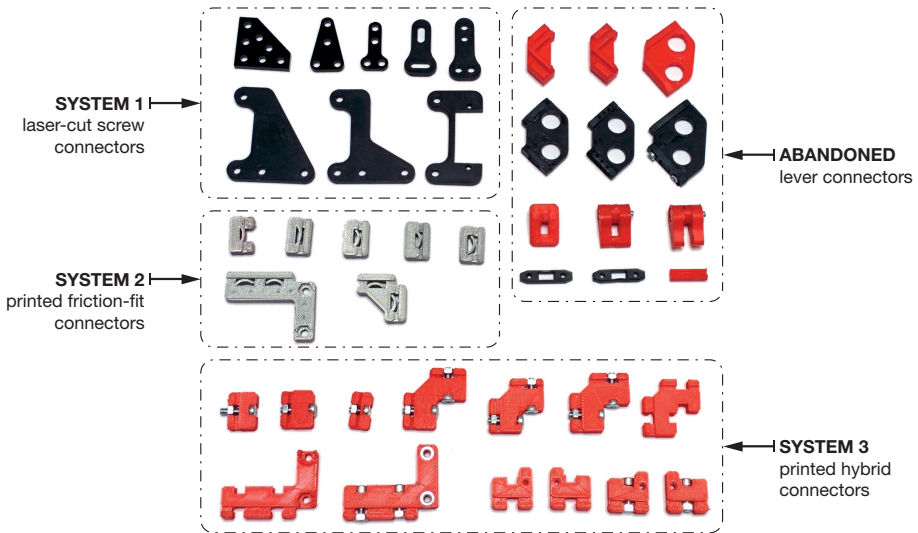
**Fig. 3.10** Design funnel – schematic overview of our design process. Adapted from Buxton (2007, p. 148), Pugh (1991, p.75).



**Fig. 3.11** Student designs from iteration one.

to come up with a type of reversible connector to guarantee the modularity of their kits. In retrospect, we think this step was the critical moment that determined the quality of the kits they designed. For this reason, we decided to focus solely on designing modular connectors in the second iteration. It is from this iteration that we generated the three systems that were used in the robotics contest. For each system, we designed a 90° corner connector (which allows for both corner- and T-connections), a servo connector, and a DC motor connector. The principle behind each system is highly adaptable, and new connectors, based on the same principle, can be easily designed to accommodate specific

sensors, larger motors, etc. The adaptability of the connectors and the continuous mounting positions offered by the aluminium beams together result in a high degree of flexibility in the three construction systems. Figure 3.12 shows some of the divergent and convergent prototypes created in iteration two. Some of the "abandoned" prototypes are also shown in this picture. For instance, one concept relied on a lever mechanism to lock the connector into place. However, forces generated by the lever caused the printed part to delaminate, and no effective solution could be found for this problem.



**Fig. 3.12** Prototypes from iteration two.

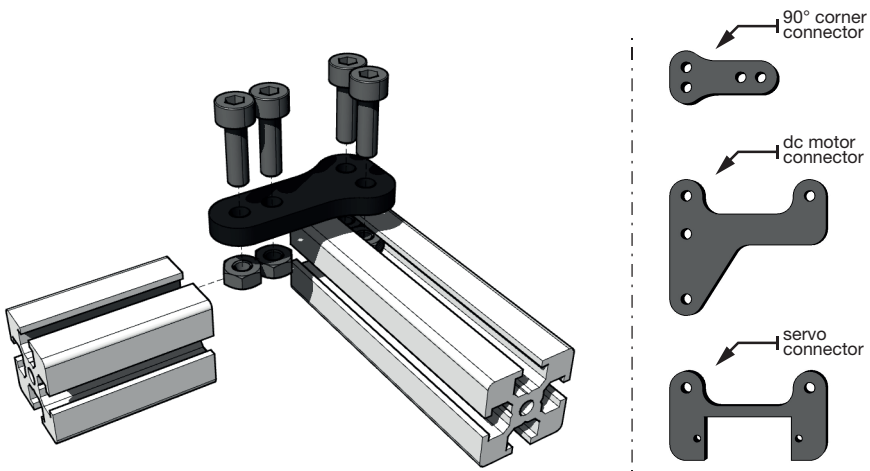
The final design files of the three systems can be found in our GitHub repository<sup>6</sup>. The parts for system 1 were laser-cut from 3 mm sheets of ABS plastic. ABS was chosen because the material is strong, but not brittle and because it can be cut easily and cleanly using a 40 mm laser cutter. Systems 2 and 3, which were designed to be 3D printed, were printed on an Ultimaker 1 using PLA plastic with a layer height of 0.1 mm and an infill density of 20%. PLA was chosen because of its ubiquity in low-end 3D printers and because the material can be printed without a heated bed. However, the pieces can also be printed in ABS, if desired. Systems 1 and 3 rely on M3 nuts and screws for their functionality. Either hex socket cap screws or cross-recessed pan screws can be used, though the former is preferred. Hex socket cap screws offer two advantages: (1) they are more durable (less prone to stripping), and (2) they can be tightened at an angle using a ball-end hex key. Since we had difficulty buying this type of screw in Argentina, cross-recessed screws were used during the contest. They work just as well, but are more cumbersome to work with in tight spaces.

<sup>6</sup><https://github.com/cesarvandevelde/RobotBlocks>

### 3.2.2.1 SYSTEM 1: LASER-CUT SCREW CONNECTORS

System 1 relies on laser cutting as the sole production technique. In this context, the main advantage of laser cutting is its speed: all the parts required for 10 teams were produced in two hours, whereas the parts for the other systems took over a week, each using 3D printing. The main design limitation of laser cutting is that materials can only be cut from one direction, so only flat shapes can be produced.

The construction system we created using this technique relies on small T-shaped gusset plates to connect aluminium beams (fig. 3.13). The gusset plates contain four holes each, which are used to screw the plates to the aluminium extrusions. Two screws are used per beam to ensure that the beams cannot rotate in respect of one another. The aluminium extrusions we chose are well suited for this application as the T-slots of the extrusions can accommodate standard M3 nuts. Alternatively, the corner plates can also be used in conjunction with the threaded holes at the end of each beam to create a connection between beams.



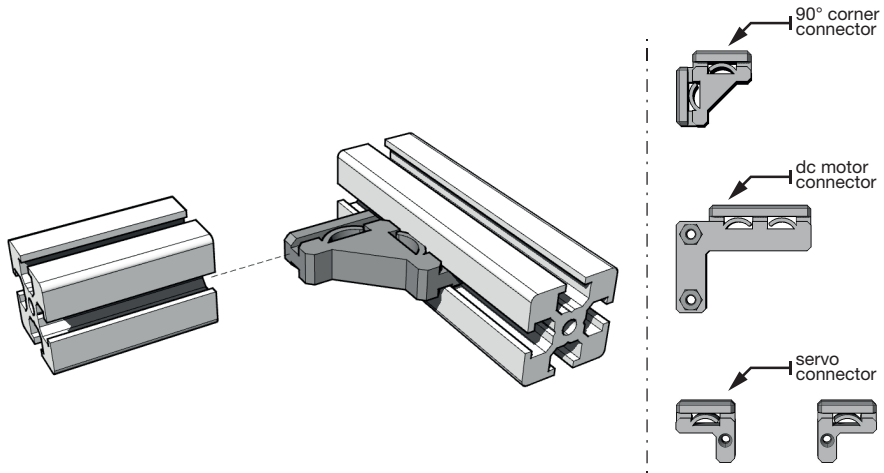
**Fig. 3.13** Laser-cut screw connector system.

To make a corner connection between two beams, two nuts need to be inserted into each beam first. Then, a gusset plate is positioned over the nuts, and screws are inserted in the four holes. Adjustment is possible by shifting the parts around in their T-slots. Finally, the connection is secured in place by tightening the screws.

This system relies on nuts and screws to create a rigid connection. While this method is reliable and low-cost, changing the construction takes some time and, therefore, limits the scope for rapid iterations of different designs.

### 3.2.2.2 SYSTEM 2: PRINTED FRICTION-FIT CONNECTORS

For the second construction system, we wanted a type of connection that is very quick to use in order to encourage quick design iterations in the robotics contest. Consequently, our second connection system relies solely on friction to connect pieces together. The corner pieces of this system have two sets of grooves that match the profile of the T-slots in the aluminium beams (fig. 3.14). To create a connection, two beams are simply slotted into a corner piece with sufficient force. The drawback of this approach is that the friction force limits the amount of force that each corner can absorb.



**Fig. 3.14** Printed friction-fit connector system.

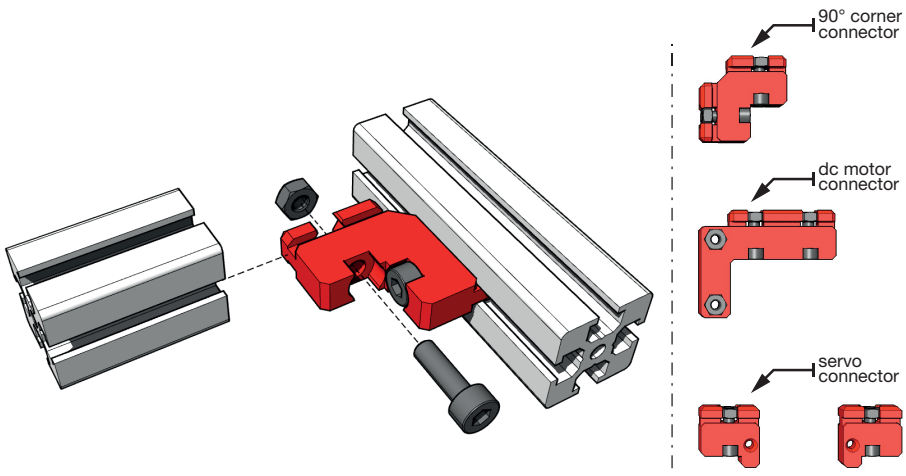
The plastic pieces of this system were designed to be printable on a low-cost 3D printer (e.g. a RepRap, or in our case, an Ultimaker). Designing for this category of 3D printers poses several limitations from which professional 3D printers do not suffer. One of the main challenges is that low-end 3D printers typically do not have a second print head to deposit support material. This means that features such as undercuts need to be carefully designed so that they are self-supporting. Parts of this system can be printed with minimal overhangs by laying them flat on their side. The only overhangs – the bottom grooves – can be printed because both edges are supported.

The second challenge we encountered while designing this building system was to find a good way of reliably creating a friction-fit connection. The tolerances of parts produced on DIY 3D printers depend on many factors, including the construction and calibration of the machine, the quality of the plastic filament used, and even the ambient temperature. As such, a simple groove with correct sub-millimetre dimensions is not a feasible way to achieve a friction fit. Our solution was to incorporate a spring-like feature in the printed parts. The purpose of this spring is to create tension between the groove of the printed part and the surface of the aluminium beam. This design allows for much wider tolerances. The spring-like feature also takes advantage of the anisotropic nature of 3D printed objects: the springs flex in the horizontal plane, which is the strongest direction in FDM parts because

it does not depend on interlayer adhesion.

### 3.2.2.3 SYSTEM 3: PRINTED HYBRID CONNECTORS

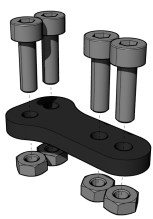
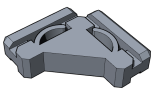
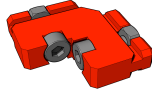
In the design of third system, we wanted to strike a balance between the strength and robustness of the laser-cut screw connectors and the ease of assembly of the printed friction-fit connectors. This hybrid system combines the groove mounting system of system 2 with the nut and screw connection of system 1. In practice, this means that users can quickly try out new ideas by sliding connector pieces into the T-slots of the beams. Once they are securely in place, the connection can be fixed by tightening the two screws.



**Fig. 3.15** Printed "hybrid" connector system.

The connector itself consists of a printed plastic part, two nuts, and two screws. The basic shape of the corner connector is similar to the one from the friction-fit system, with grooves to accommodate the T-slot channels of the aluminium beams (fig. 3.15). Additionally, each side has a circular hole ending with a hexagonal cut-out, which holds a nut and screw in place. The nut sits in the hexagonal cut-out and is positioned in line with the ridge that slides into the T-slot channel of an aluminium beam. This connector can be inserted into the extrusions with the fasteners already in place, resulting in a substantial speed gain over the laser-cut screw connectors.

As with the previous building system, a number of properties of low-end 3D printing are utilized: the parts can be printed without the use of support material, they are designed with wide tolerances in mind, and they can be oriented so that inter-layer forces are reduced to a minimum. Table 1 shows a summary of the properties of the three systems.

	System 1	System 2	System 3
			
Production technique	Laser cutting	3D printing	3D printing
Material	3mm ABS sheet	PLA	PLA
Production time per piece	<1 min.	15 min.	15 min.
Connection method	screw	friction	friction + screw
Extra hardware per connector	4 M3x6 screws 4 M3 nuts	none	2 M3x10 screws 2 M3 nuts

**Table 3.1** Comparison of the three building systems

3.2.3 MEASURING USABILITY, AFFECTIVE APPRAISAL, AND FUNCTIONALITY

The main objective of our evaluation is twofold. Firstly, we wanted to determine which of the three systems is most appropriate for use in educational robotics. Secondly, we wanted to establish the next steps to further improve that system. To this end, we focused on three key aspects during our evaluation process. These aspects are (1) the usability (i.e. *are the blocks easy to use?*), (2) the affective appraisal (i.e. *what is the perceived emotional value for users?*), and (3) the functionality (i.e. *how versatile are they, how well do they perform?*).

The tools we selected to measure the above-mentioned aspects were subject to a number of constraints. We chose tools that are short and quick to fill out, that are unambiguous, and that can be completed through an online survey tool. Consequently, we chose the following three tools: system usability scale ( $M_1$ ), Pick-A-Mood ( $M_3$ ), and AttrakDiff ( $M_2$ ). More information on these measurement tools is given in section 2.1.1.1.

Additionally, we also asked respondents to indicate their age, gender, and the colour (which corresponds to the type) of the building blocks they used. Finally, we provided two open text areas where we asked what they liked, and disliked, about the system ( $M_6$ ).

The questionnaire was given at the end of the event, after the final contest. We asked that only the persons who actively participated in the mechanical construction process complete the survey, as the survey relates to the robot building blocks. In most teams, one person was responsible for building the physical embodiment, while the others focused on the electronics and the programming. Consequently, in those cases, only one questionnaire was completed per team. In rare cases where multiple participants worked on the mechanics, they each filled in a separate survey.

While our questionnaire certainly measures usability and affective appraisal, functionality is not measured as explicitly. For that reason, we also conducted an expert evaluation. We asked six experts – teachers and coaches who are frequently involved in educational robotics – to participate in a two-part study. The experts were chosen for both their experience in teaching and their knowledge regarding the design of robotics. In the first part, they were asked to indicate which aspects they consider when evaluating robots. They were then shown short (less than 1 minute) video clips of 17 robots built during the CERobotics contest ( $M_8$ ), and were asked to give each of them a score between 1 and 9 ( $M_4$ ). For the second part, the experts were given the opportunity to experiment with the three different systems. Subsequently, they were asked to rank the systems in their order of preference, and to write down any additional comments they had ( $M_6$ ). While evaluation through video files certainly has its limitations, we feel that this approach, in combination with the responses from the open questions, do allow us to gain an insight into the functionality of each robot construction system.

### 3.2.4 RESULTS

Of the 86 students and 35 teachers (121 participants total), 37 participants indicated they were actively involved in the mechanical building process and were asked to complete our questionnaire. This corresponds to a ratio of 30.6%. As mentioned above, one person per team completed the survey in most cases. At the start of the project, each group was assigned a building system in order to achieve an approximately even distribution (resp. 12, 12, 13) of the three systems. However, early on in the build process, several of the groups using the laser-cut screw system expressed their frustration with this system. They were therefore allowed to switch to a different building system of their choice, which resulted in a disproportionate distribution, where the laser-cut screw system was severely underrepresented as compared to the other systems. Four of the questionnaire respondents used the laser-cut screw system, 15 used the friction-fit system, and 18 used the hybrid system.

The average age of respondents is 18.5. However, this average is skewed because the groups consisted of a majority of secondary school students ( $n = 32$ ) supplemented with a small group of teachers ( $n = 5$ ). Average age of students was 16.1 ( $\sigma = 1.65$ ); average age of teachers was 33.0 ( $\sigma = 9.19$ ). Gender data show a male majority, with 27 male participants and 10 female participants. However, gender ratio skewness is not uncommon in robotics contests (Johnson, 2003; Milto et al., 2002).

#### 3.2.4.1 SYSTEM USABILITY SCALE

The results of the System Usability Survey showed an average SUS value of 80.8 ( $\sigma = 14.6$ ) for all three systems combined. Bangor et al. (2008) calculated the average SUS value of 206 studies to be 69.69. A one-sided t-test at a significance value alpha of 0.05 indicates that the average value of 80.8 is statistically significantly different from the baseline value of 69.69, ( $t = 4.621$ ,  $p < 0.05$ , 95% CI of the difference = [6.24, 16.001]).

To investigate differences in SUS scores among the three systems, a one-way ANOVA was performed. Inspection of the boxplots and the Kolmogorov's test ( $p = 0.068 > 0.05$ ) suggests normality of the data. Moreover, the assumption of homogeneous variance was confirmed by Levene's test ( $p = 0.261 > 0.05$ ). Based on the one-way ANOVA, with a significance level of 5%, ( $F = 0.007$ ,  $p = 0.993 >> 0.05$ ), we can state that there is not enough statistical evidence of difference in mean value of SUS among the three systems. We also performed a Kruskal-Wallis test, which confirms the result of the one-way ANOVA analysis ( $p = 0.97 >> 0.05$ ). This result is in contrast with our experiences early on in the project, where several groups switched from the laser-cut screw connector system to another system. A possible explanation is that only the users who were satisfied with the system remained, which would explain why no difference in usability is detected.

Indeed, two users of the laser-cut screw connector system indicated in the open questions that they had difficulty joining beams with this system. We think this is because the system is particularly sensitive to the order of assembly, because the nuts need to be inserted into the channels of the beams in advance. The main problems reported by users of the friction-fit system is that connection pieces require too much force the first time they are used, and that they are prone to loosening while in use. Users of the hybrid bricks reported only minor issues, such as the hexagonal openings for nuts being too small. In all three groups, users remarked that they would like a larger variety of different pieces. We suspect that the usability of systems 1 and 3 would have been slightly higher if hex socket cap screws were used instead of cross-recessed pan head screws, as described earlier.

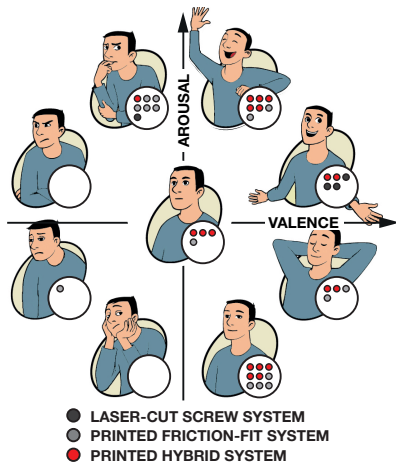
### 3.2.4.2 PICTORIAL MOOD REPORTING INSTRUMENT

The second part of our survey uses the Pictorial Mood Reporting Instrument (Desmet et al., 2012; Vastenburg et al., 2011) to measure the overall mood participants experienced during the construction process of the robot. The facial expressions provided by this tool can be arranged on two axes, comparable to Russel's circumplex model of affect (Russel, 1980). These axes are valence (pleasure - displeasure) and arousal (high energy - low energy). Figure 3.16 shows the moods reported by participants plotted on these two axes. Fisher's Exact Test (Fisher, 1922) indicates that there is not enough statistical evidence ( $p = 0.59 > 0.05$ ) to claim that there is a relationship between the type of building system used and the mood reported by the users. We think the low sample size is partially responsible for this. If we take the laser-cut connector pieces ( $n = 4$ ) out of the equation, and cluster the moods in positive, neutral, and negative brackets (cfr. the valence axis in the circumplex model of affect (Russel, 1980)), we do see a slight correlation ( $p = 0.047 < 0.05$ ) between the building system and the valence of the reported mood, with the hybrid connectors performing slightly better.

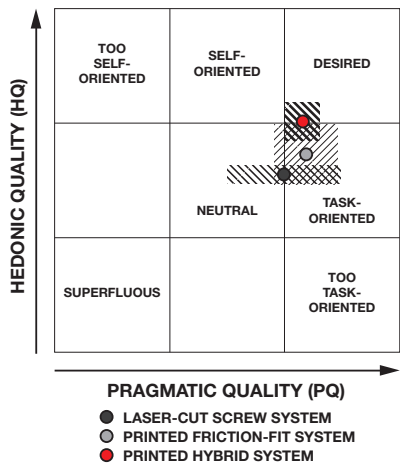
### 3.2.4.3 ATTRAKDIFF

The third part of our survey consists of the AttrakDiff questionnaire (Hassenzahl et al., 2003). The data was processed by means of the AttrakDiff online tool. Figure 3.17 shows





**Fig. 3.16** PMRI results, each dot represents the mood of one respondent.



**Fig. 3.17** AttrakDiff results, the hatched area represents the confidence interval of each system.

the position and confidence rectangle of hedonic and pragmatic qualities of the three systems. Overall, we can say that the hybrid system scored better than the friction-fit system, which, in turn, scored better than the screw connector system. It should be noted that the large confidence interval of the laser-cut screw connectors is again a consequence of the low number of participants compared to the other two systems. The results from the AttrakDiff evaluation are in line with the feedback from the open questions and with our own subjective assessment. Although these results indicate that the hybrid system is the best of the three, there is still room for improvement.

### 3.2.4.4 EXPERT EVALUATION

As a final part of our study, we performed an expert evaluation with six experts. In the first part of this evaluation, the experts were asked: “In your opinion, what criteria are important for grading mechatronics projects such as a robotics contest?” (Q1, fig. 3.18). The experts were then shown video clips of 17 different robots and were asked the following question: “Using the criteria and their importance you specified in the previous section (Q1), how would you rate the robots shown below?” Each robot could be rated individually using a 9-point scale, with 1 being the lowest and 9 being the highest rating. A 9-point scale was chosen to allow for more granular reporting than 5-point scales, while still offering a neutral position. Of the 17 robots, four were built using the laser-cut screw connector system, seven using the printed friction-fit system, and six using the printed hybrid system. The results of Q2 (table 3.2) show an average score of 6.04 for robots built with the screw connector system, 4.52 for those built with the friction-fit system, and 6.75 for robots built using the hybrid system. While the sample size of this study is low, data does suggest that the use of the hybrid system tends to lead to better scoring robots.

System	1: laser-cut screw connectors				2: printed friction-fit connectors								3: printed hybrid connectors					
Robot	H	M	O	P	A	B	C	E	F	J	K		D	G	I	L	N	Q
E <sub>1</sub>	9	2	9	2	8	2	6	2	2	2	2		6	2	8	6	9	8
E <sub>2</sub>	9	5	8	1	1	1	8	1	8	1	5		5	1	8	8	8	8
E <sub>3</sub>	8	5	8	5	2	2	7	3	6	4	6		5	3	7	7	8	8
E <sub>4</sub>	9	3	8	2	4	3	7	3	8	4	7		6	4	7	8	9	9
E <sub>5</sub>	8	5	8	3	4	4	6	4	7	5	6		6	6	9	8	8	9
E <sub>6</sub>	9	6	9	4	4	3	7	3	8	7	7		6	5	7	5	7	9
Avg.	8.67	4.33	8.33	2.83	3.83	2.5	6.83	2.67	6.5	3.83	5.5		5.67	3.5	7.67	7	8.17	8.5
	6.04				4.52								6.75					

Table 3.2 Q2 – Robot ratings

In the last part of the expert evaluation (Q3), the experts were given samples of the three systems to experiment with. They were then asked to rank the systems in order of preference using three drop-down menus. Results of Q3 are shown in table 3.3. The hybrid system was ranked first the most (three times), followed by the friction-fit system (twice), and then the laser-cut system (only once). The experts frequently praised the ease of use of systems 2 and 3, but noted that the friction-fit system will probably loosen over time. In their comments, the experts also commended the laser-cut system for its simplicity and strength, remarking that it is a very cheap part to produce.

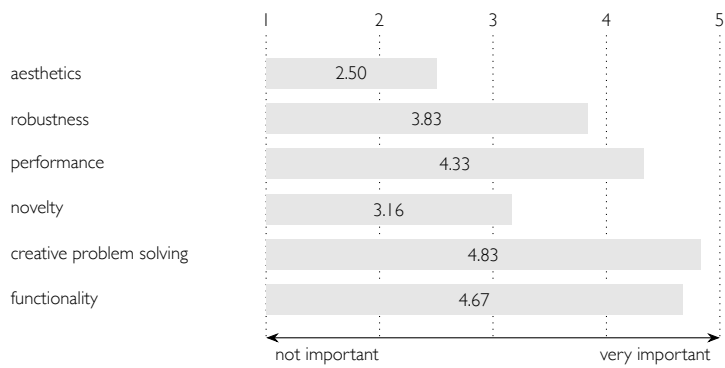


Fig. 3.18 Q1 – Experts' criteria for grading robots.

3.2.5 SUMMARY

The results constitute a first step towards an open, DIY construction kit for small-scale robotics. We believe such a system has the potential of greatly complementing other efforts in educational robotics by providing a low-cost ( $G_9$ ), “hackable” ( $G_1$ ) platform for building the physical embodiment of small robots.

We believe that the use of T-slotted aluminium beams helped us greatly toward this goal:

System	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	E <sub>4</sub>	E <sub>5</sub>	E <sub>6</sub>
System 1: Laser-cut screw connectors	2 <sup>nd</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	2 <sup>nd</sup>	1 <sup>st</sup>	3 <sup>rd</sup>
System 2: Printed friction-fit connectors	3 <sup>rd</sup>	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	2 <sup>nd</sup>	1 <sup>st</sup>
System 3: Printed hybrid connectors	1 <sup>st</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	1 <sup>st</sup>	3 <sup>rd</sup>	2 <sup>nd</sup>

**Table 3.3** Q3 – Systems ranking by the experts

they are low-cost, strong, and can be used to connect many different types of components. However, the downsides of this approach include the weight (the aluminium beams are heavier than their plastic counterparts) and their appearance (robots built with this system can be bulky and technical looking).

Of the three construction systems presented in this paper, we think the hybrid system is best suited to the context at hand. This conclusion is also supported by the results of our questionnaire. While feedback indicates that the friction-fit system is easy and pleasant to work with in the assembly phase, it does cause problems when the robot is used, due to failures of the connection under excessive force. On the other hand, the laser-cut screw system can be manufactured quickly and provides strong, firm connections. However, these connections are slow and difficult to use, resulting in frustrated users and hindering the state of flow. The hybrid system strikes a balance between the strength and rigidity of a bolted connection and the ease-of-use of a friction-fit connection. Users can quickly try out different configurations by slotting the connectors in and out of the beams, and once they are happy, connectors can be locked into place by tensioning a single screw per side. This allows for a better match between skills and challenges, leading to a positive state of flow ( $G_6$ ) for the user. The hybrid system offers a good balance between technical performance, ease-of-use ( $G_2$ ), and cost ( $G_9$ ). The insights gained from the hybrid system were later integrated in the latest version of the Opsoro platform (section 3.5.3).

A common point of criticism that applies to the three systems is that users want a larger variety of building blocks. We only provided three types of building blocks for this project: a 90° corner connector, a DC motor connector and a servo connector. While just these three types are adequate for building a firefighting robot, we recognize that this basic selection of blocks may have been limited, though it did force the users to use creative thinking to design solutions with the limited set of building blocks ( $G_7$ ). Our first set of connectors provides only static connections, although we have every intention of creating components that allow for moving mechanisms, such as hinges, wheels and gears. Additionally, because the design files of the three systems were released as open source hardware, motivated users can extend the systems with custom-designed components ( $G_1$ ,  $G_{10}$ ).

While on this subject, we would also have liked to involve the students in the manufacturing process of the building blocks. As (Blikstein, 2013a) showed, FabLabs and digital fabrication offer many STEM-related teaching opportunities ( $G_8$ ). However, due to lo-

gistical challenges, this was not possible for our robotics contest in Argentina. Instead, all components were manufactured and kitted beforehand.

As a final point, we would like to continue improving our evaluation method. While we gained valuable insight through the use of questionnaires, we did still encounter some problems. We purposefully selected evaluation tools that are quick to fill out and avoided too many open questions, but we still noticed that some participants found the questionnaire too long. As an alternative, we would like to experiment with periodic evaluation forms, where we ask participants to fill out a very short survey at the end of each session.

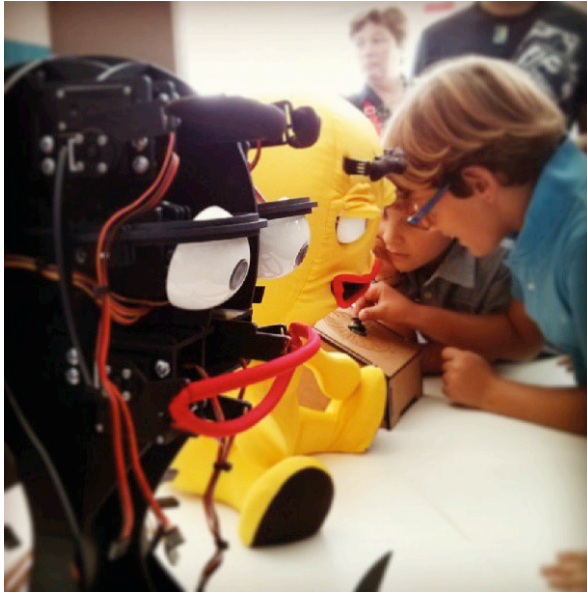
### 3.3 ONO – GENERATION I

The Ono project finds its origin in our experiences with conventional social robots, chiefly the robot Probo (Saldien, 2009). Experiments with the Probo have shown a discrepancy between what the designers envisioned as important functionality, and what functionality was actually used by therapists during experiments. The robot had been conceived as an advanced research platform of which only one copy would be built. As it turns out, many interaction experiments with children required only basic robot functionality (Pop et al., 2013; Vanderborght et al., 2012). Experiments were also at times hampered by practical issues such as malfunctions, broken components, setup and calibration, and transportation difficulties. In this, we recognized an opportunity for a low-cost and low-tech hackable social robot.

This insight prompted us to design a new, simplified social robot, as we found that others were also dealing with similar issues. Our goals for the design of the Ono robot were to create a robot that is inexpensive, reproducible, modular, easily repairable, and easily transportable. Many of these challenges were met by taking cues from contemporary DIY paradigms such as the maker movement and the open source hardware movement, as described in chapter 1. We believe that low-cost open source social robots provide advantages that current high-end robots do not have: they are well suited for large-scale studies and they are accessible for students and hobbyists.

The design of Ono is aimed primarily at children, with a focus on therapeutic applications such as the treatment of autism spectrum disorder. The reason we chose this target group is twofold: (1) there is a large demand for this type of robot for treatments of developmental disorders such as autism (Cabibihan et al., 2013) and (2) other social robots (e.g. Kozima, M. P. Michalowski, et al. (2009), Dautenhahn et al. (2009), Saldien et al. (2010), Metta et al. (2008)) exist within this segment of therapeutic robots for children, providing us with a point of reference. By narrowing our focus to this application area, we gain more specific information about the usage context of the robot, which is helpful in the design process.

The use of digital manufacturing techniques is ideally suited for an iterative design process. Consequently, Ono's version history is much more a continuous spectrum of small changes (just like "commits" in software version control systems), rather than large, dis-



**Fig. 3.19** Children interacting with Ono v1.1 through the control box

crete product releases. Nevertheless, we distinguish the following versions within the first generation of Ono:

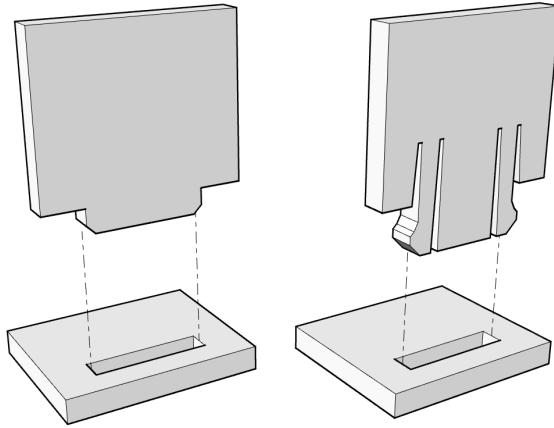
- **ONO V1.0** – The first functional version of Ono. This milestone is preceded by experimentation with materials and production techniques, multiple iterations of mechanisms for facial animation, as well as character design and modeling. The design was split into two different prototypes: the first being a fully working “naked” robot and the second a static proof-of-concept demonstrating the materials and techniques to be used for the robot’s skin.
- **ONO V1.1** – The most important change in this version is that it combined the two previous prototypes into one fully integrated robot. While the two prototypes of v1.0 were designed with integration in mind, many small tweaks were needed to achieve this goal. The order of operations to attach the skin over the skeleton proved especially challenging.

In addition to the design changes made to enable integration, we also transitioned from using PS to using ABS for all laser-cut parts. ABS plastic is stronger and a lot less brittle than PS. The material also cuts much better, eliminating burrs, reducing post-processing steps, and leading to a better part finish.

- **ONO V1.2** – The first change of this version relates to the construction of the robot’s laser-cut frame. Whereas the previous version relied upon friction to hold slot-and-tab connections in place, v1.2 replaced the plain tabs with cantilevered

snap, providing a positive locking force. Figure 3.20 shows the difference between old-style and new-style connectors.

In this version, we also experimented with a new Arduino-based controller integrated into the robot's body. This is a break from the previous setup, where processing happens in a separate control unit. More detail on this controller will be described in section 4.2.1.



**Fig. 3.20** Changes in the slot-and-tab construction system between v1.1 (left) and v1.2 (right)

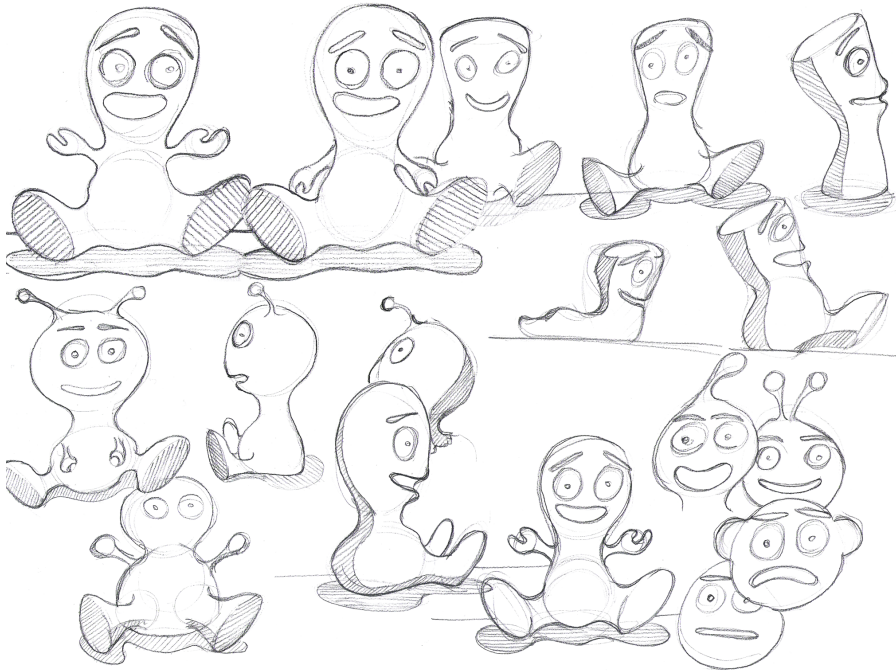
The first generation of design iterations did not yet embody all aspects of toolkits, as described in section 1.1.5. Still, primitive elements, such as the snap connectors and the modular elements, have been part of the design since the first version.

The robots in this generation are characterized by a focus on mechanical design aspects. The scope of the electronics and software implementation is limited to the bare necessities required for basic animation of facial expressions. The main contributions of the first generation prototypes is that they served as a test bed for experimenting with materials and techniques, and allowed for a preliminary validation of functionality scope and character design decisions.

### 3.3.1 CONCEPTUAL DESIGN OF THE EMBODIMENT

The embodiment design of the Ono is the result of three categories of constraints: (1) requirements related to the target audience of the robot, (2) technical requirements, and (3) practical requirements related to the usage in a therapeutic context. This section gives an overview of the rationale behind each of these constraints, and briefly explains how these constraints have shaped the visual appearance of the robot.

We decided to limit actuation to the face of the robot and keep the rest of the body non-moving. Actuation of facial features can be achieved with comparatively small, inexpensive



**Fig. 3.21** Exploration sketches for the embodiment design

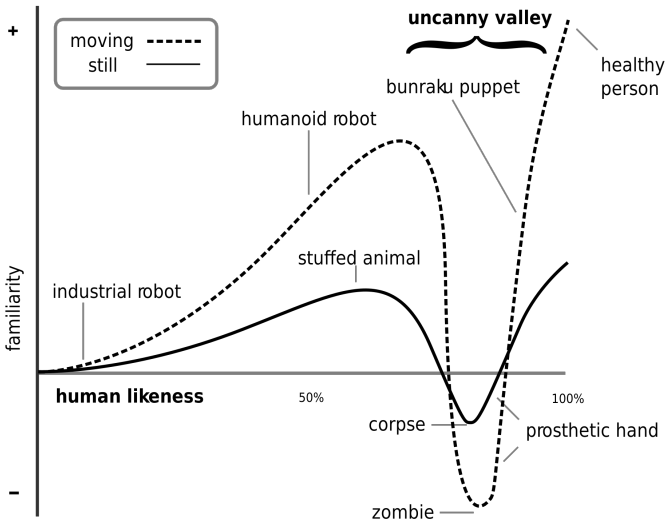
actuators, whereas full body motion necessitates the use of high-performance motors and drive systems, driving the cost up considerably. Multiple studies and applications in HRI make use of facial expressions since people rely on face-to-face communication in daily life. The face plays a very important role in the expression of character, emotion and/or identity (Cole, 1998). Mehrabian (2008) showed that only 7% of affective information is transferred by spoken language, that 38% is transferred by paralinguistics and 55% of transfer is due to facial expressions. Facial expressions are therefore a major modality in human face-to-face communication, especially in fields such as Robot-Assisted Therapy (RAT), where emotions play a crucial role in the communication process.

The entire robot is covered in a soft foam and textile skin in order to attain a soft and inviting appearance for children, as well as to protect the internal components from damage. We wanted to move away from the image of robots as mechanical devices made of hard plastics and metals, and instead aim for a non-threatening image reminiscent of stuffed animals and teddy bears. The rationale behind this choice is in line with that of similar robots such as *Paro* (Kidd et al., 2006), *Probo* (Goris et al., 2011) and the *Huggable* (Stiehl et al., 2009).

Because of the process we chose for the production of *Ono*'s soft foam skin layer, the types of geometry we could produce was limited to simple, smooth shapes. This is reflected in the final design of the robot, which could be described as a "blobject"; an object composed

smooth curves and lacking sharp edges. This is necessary because the three-dimensional foam layer is composed solely out of many flat pieces of foam, and more detailed forms would require more foam parts.

The unrealistic, icon-like appearance is not only the result of technical limitations. It is also a conscious decision made to avoid the effects of the Uncanny Valley (fig. 3.22), as described by Mori (1970). The hypothesis of the Uncanny Valley is that the affinity between a human being and an object increases in proportion to the human likeness of that object. This effect continues up to the point where the object (e.g. a robot) is almost, but *not quite* human. In this region, named the Uncanny Valley, the eeriness results in a strong, negative response. In our case, we deliberately chose to stay on the left side of the Uncanny Valley, opting for a character-like appearance over a near-human likeness.



**Fig. 3.22** The Uncanny Valley. Adopted from MacDorman (2005).

The size of the Ono is a direct consequence of one of the most important practical concerns, namely that the robot should be easy to travel with. The robot is sized so that it fits inside the maximum dimensions for an airline carry-on bag, 22 cm × 35 cm × 56 cm. This is important for demonstrations at conferences, workshops, and collaboration with international partners.

The robot has a disproportionally large head to draw attention to its face, making its facial expressions more noticeable. Ono is also posed in a sitting position to improve stability. As a consequence of its size and pose, children can interact with the robot at eye height when the robot is placed on a table.

The color yellow was chosen because of the association between yellow and positive emotions, as described by Kaya and Epps (2004). The yellow skin color also provides sufficient



contrast to the facial features, making them easily distinguishable.

Finally, we chose for a generic, non-descriptive appearance for robot instead of opting for a more fleshed-out character with distinct features. The reasoning behind this is that embodiment serves as a sort of blank canvas onto which a character can be built through the use of attributes, a bit like the Mr. Potato Head toy. One idea here was to embed RFID tags in the attributes, allowing the robot to automatically change the behavior based on its attributes.

The design requirements for the conceptual design of the robot are summarized in table 3.4. Figure 3.21 shows some of the doodles and sketches from the concept design phase. Figure 3.23 shows how the elements from the sketches were combined and incorporated into a 3D model. This model served as a basis from which the detailed CAD drawings of Ono were created.

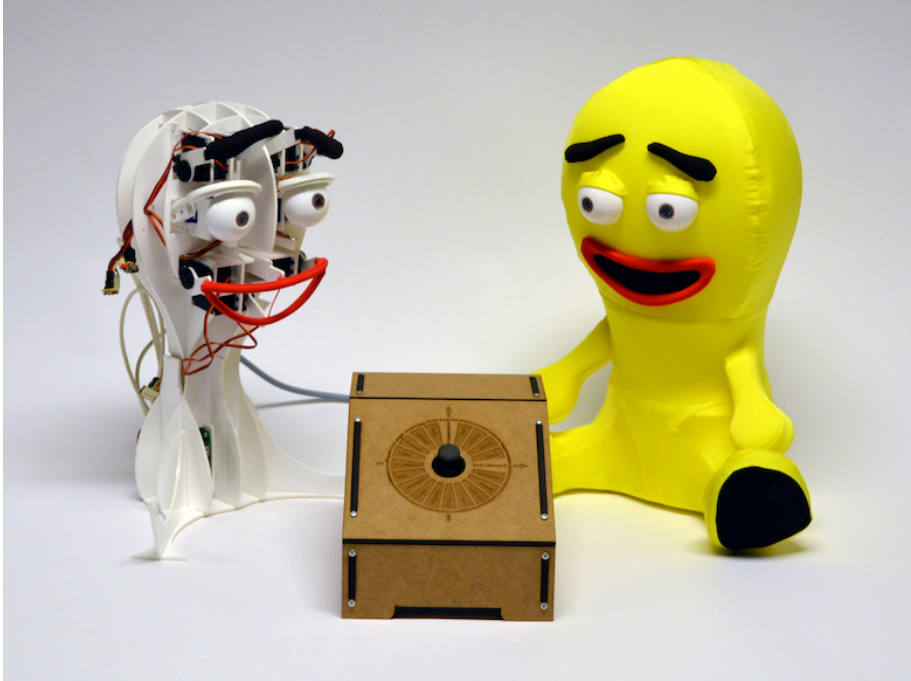


**Fig. 3.23** Rendered image of the Ono outer skin surface model

	Target audience (children)	Technical requirements	Practical requirements
Face only		✓	✓
Soft & huggable	✓		✓
Simple shape		✓	
Unrealistic appearance	✓		
Fits in hand luggage			✓
Large head	✓		
Seated position	✓	✓	
Yellow	✓		
Generic appearance	✓		✓

**Table 3.4** Summary of concept design requirements

### 3.3.2 CONSTRUCTION



**Fig. 3.24** The Ono v1.0 prototype with control unit

Different from many other robots in HRI research, Ono was designed with low-cost materials and DIY techniques in mind, with the goal of creating a low-cost social robot that can be produced using equipment that is commonly found in FabLabs. The cost of materials required to build one robot is around 310 EUR. This cost may vary depending on location and on what components the user already owns. Table 3.5 provides a rough breakdown of the costs.

The design of the robot relies upon laser cutting as the main production technique. The main advantages of laser cutting are that (1) it is fast, (2) the files can be edited easily, (3) it is well suited for larger components and (4) the machine is reasonably easy to operate, requiring no artisanship. As a self-imposed constraint, we restricted the design to using only laser-cutting and off-the-shelf parts in generation one, because we did not want to introduce more machinery/equipment requirements. This proved to be a challenge for the design of the eyes, which require many small and intricate components.

Both the hard, mechanical parts and the soft, protective foam covering are cut using a laser cutter. The structural parts are made from 3 mm thick plastic sheet material. Initially, we used white PS plastic, though from v1.1 onward, we switched to black ABS plastic due to its superior mechanical properties and better part finish at a similar price point. The soft foam covering is made from flat sheets of flexible PU foam.

Figure 3.24 shows a picture of the v1.0 prototype of the robot. The left side shows the inner workings of the robot, a proof-of-concept of the outer skin is shown to the right, and the control interface box is shown in the middle. We distinguish four key elements in the design of the robot:

- *The frame* – The frame of the robot is made from laser-cut interlocking cross-sections, which slot together to form a sturdy structure. The pieces of the frame are attached to one another using slot and tab connections. The frame has openings to accommodate the different modules and has slots and holes to secure the wiring to the frame using zip-ties.

A 3D model of the desired appearance of the robot was used as the starting point for the frame. After accounting for the thickness of the foam covering, this model was sliced into multiple cross-sections. These sections were then used as the basis for the skeletal structure of the robot. The frame design was finished by adding features such as connectors, slots and holes to the cross-sections. Currently, we generate these slices manually, however, future work includes programming a CAD tool or plugin to automate this process.

- *The modules* – The facial features of Ono are divided into modules. Each module consists of a set of related actuators and structural parts, forming a higher-level building block. Modules are connected to the frame using cantilever snap connectors. We chose to group components into modules because it facilitates repairs, because they can be reused in new robots and because they allow us to improve facial feature mechanisms independently from the rest of the robot.

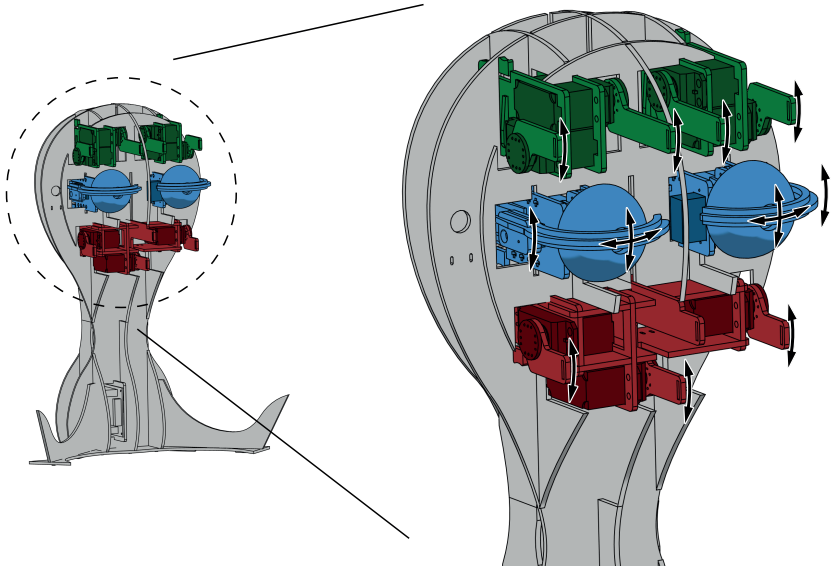
Figure 3.25 shows the modules (highlighted in color) embedded in the skeletal frame. The arrows indicate the degrees of freedom of the Ono's face. The generation one prototypes have three types of modules: two eye modules (blue), two eyebrow modules (green), and one mouth module (red).

- *The skin* – The frame and the modules are wrapped in a protective cover made from polyurethane foam. This gives the robot a soft exterior for interaction with children and protects the inner components from potential damage. This soft foam covering is in turn covered by a sewn suit made out of elastic fabric, covering up the inner components and providing a visually attractive appearance.

The manufacture of the soft foam padding proved difficult. Initially, we experimented with molding techniques to produce this flexible foam cover. This proved to be unsuitable: creating large molds is labor-intensive and the flexible PU resins are very sensitive to temperature and humidity conditions. Our solution was to recreate the three-dimensional foam cover out of many flat laser-cut pieces of foam. The 3D model of the intended appearance was divided it into regions. These regions were then flattened in software in a way that minimizes total amount of distortion. The resulting shapes were then laser-cut out of foam and then sewn together over the frame, resulting in a 3D foam shape. The technique requires that the shape of the model is smooth and continuous, so that the body can easily be split in two-dimensional patterns. The same 3D flattening process was also used to create the patterns for the textile skin.

- *The control unit* – The first generation prototypes are controlled from a separate control box. In addition to the interface, this control unit also contains the power supply and the microcontroller driving the robot. As these prototypes were designed primarily as technical benchmarks for the mechanical design, much less attention was paid to the software side of the robot. Consequently, these robot contained only the bare minimum of interactivity, offering just enough to be able to test the mechanical engineering.

The front of the control box has a joystick positioned in the center of an emotion chart, allowing the user to select the desired emotion directly. This chart is based upon the circumplex model of affect (Russel, 1980), a continuous two-dimensional emotion space. The  $X$  and  $Y$  positions of the joystick are mapped directly to the dimensions of this model, and are used to drive facial expressions. The internal microcontroller of the control unit reads the joystick position and calculates servo positions. The position data is sent through a cable to the servo controller inside the robot. This cable is also used to provide power to the robot.



**Fig. 3.25** CAD model showing construction and DOFs of Ono. The servo actuators are shown in a darker color.

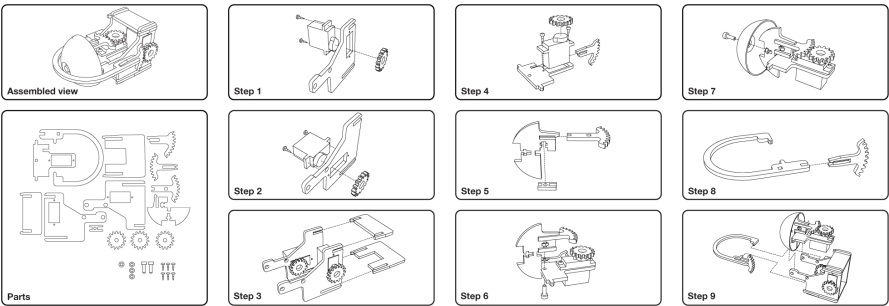
### 3.3.3 REPRODUCTION OF A ROBOT

One of the project goals is that Ono can be built without the aid of paid experts or professionals, using common materials and using easy accessible production techniques. The process of replicating the robot is made up out of two parts. The first phase consists of gathering the necessary off-the-shelf parts as well as manufacturing all custom parts. The second phase entails assembling these parts into a meaningful artifact.

Description	Cost
3mm polystyrene sheets	20 €
20mm polyurethane foam	5 €
Arduino Uno microcontroller	25 €
SSC-32 servo controller	40 €
PC power supply	25 €
RC servos	80 €
Nuts, bolts, cable ties	10 €
Connectors and electric components	30 €
Textile supplies	30 €
Laser cutting cost	45 €
Total	310 €

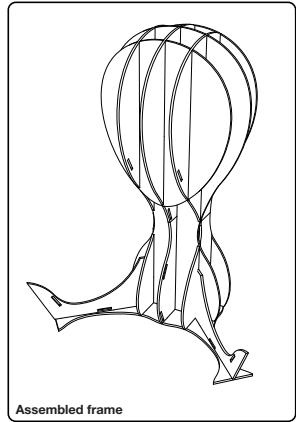
**Table 3.5** Cost breakdown of the v1.0 prototype

The focus of our investigation is on the second phase. In our opinion, the usage of digital manufacturing techniques by novices, especially 3D printing and laser cutting, has already been sufficiently explored in literature (Blikstein, 2013a; Gibb, 2014; Martin, 2015; Mueller and Baudisch, 2015). Additionally, locations that offer access to digital manufacturing facilities to the general public usually have a system in place to instruct novices on how to use the machines. For instance, many FabLabs employ a system of mandatory instruction workshops before a machine can be used.



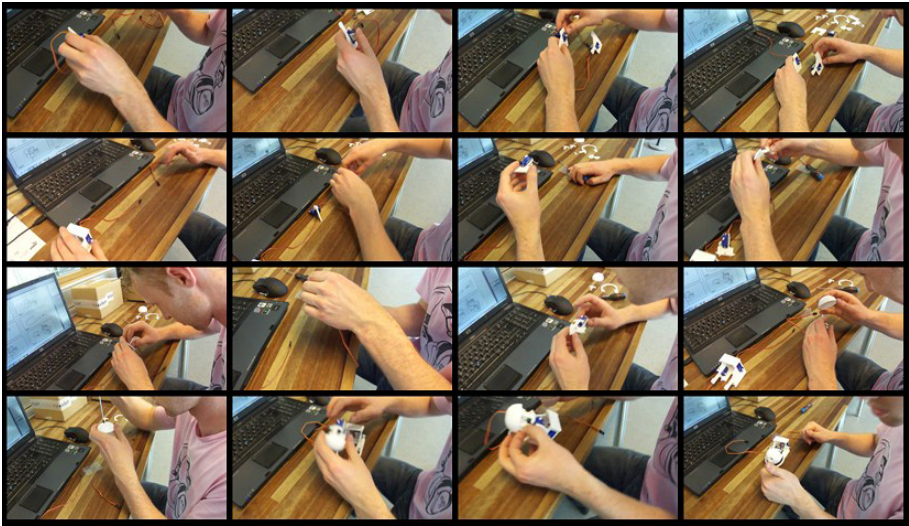
**Fig. 3.26** Assembly instructions for the Ono v1.0 eye module

On the other hand, communicating assembly instructions to non-experts can often be a difficult prospect, especially when the subject matter is complex and when face-to-face instruction is not feasible. The assembly of IKEA furniture is a classic example of this problem; though in this case, the components are not created by the user and the design is optimized to minimize the number of assembly steps. The dissemination of open hardware build instructions is especially problematic because the artifacts tend to be much more complex, and capturing all the required assembly instruction in digital files can be difficult and time-consuming.



**Fig. 3.27** Line drawing of the assembled frame

Irwin et al. (2015) illustrate some of the difficulties encountered in replication of open source hardware by novices. In the experiment, participants assembled a RepRap 3D printer (Jones et al., 2011) over the course of a three-day workshop. Here, a wiki-based instruction manual served as the primary source of documentation. Though the workshop was self-paced, experts were also present to aid participants when help was needed.



**Fig. 3.28** Participant assembling the eye module.

To investigate the potential difficulties in the assembly process of the Ono, we conducted a short pilot study. Our experiment was much more limited in scope than that of Irwin et al. (2015), however we opted to ask our participants the assembly steps *without* the aid of an expert. The experiment was divided into two tasks:

- For the first task, participants were asked to assemble the frame of the robot. They were given printed copy of a line drawing representing the intended result, they were not given step-by-step instructions. The line drawing is shown in figure 3.27.
- The second task comprised assembling one eye module. Toward this end, participants were given the required parts, simple tools, and a four-page instruction manual, shown in figure 3.26.

The pilot study was performed with five participants, aged 18-50 years old. Video recording was used as the primary means of data capture ( $M_8$ ), as shown in figure 3.28. The think-aloud protocol (Fonteyn et al., 1993) was employed during the assembly process, meaning that participants were asked to verbalize their thought process while they were assembling components. Immediately after completing the two tasks, A short follow-up interview ( $M_7$ ) was performed, consisting of the following questions:

- How would you describe the overall difficulty of the two tasks?
- What were the most difficult steps?
- How should the documentation be improved?

The tests showed that the visual instructions work reasonably well to guide inexperienced users through the assembly process. Users noted that while the drawing of the completed eye module looks very complex, the actual assembly process is greatly simplified by the step-by-step instructions. One user compared the module kit to a set of LEGO bricks and that completing the assembly of the eye module gave him a sense of satisfaction, potentially hinting at the IKEA effect (Norton et al., 2012).

The experiment also revealed a number of problems for both the instructions and the parts themselves. The instructions should include more color to identify new or dissimilar components. Each component should also be labeled with an identifying number and required tools – if applicable – should be shown in each step. At times, problems arose when similar-looking laser-cut parts were interchanged, or when parts were assembled in a mirrored position.

From generation 2 onward, a number of changes were enacted to combat the problems encountered during this experiment. To start, the instruction format was switched from an IKEA/LEGO-like line drawing format to an online wiki-based instruction manual with each step documented with a photograph and a short text descriptions. The line drawings proved too time-intensive to create and did not convey all the information that a good picture could convey. The wiki allows users to improve the documentation, and also keeps track of the document's version history.

In later generations, each laser-cut part was given a unique part number, engraved into the surface of the part. The part number also identifies the subassembly to which the part belongs, as well as the robot version number. The part numbers are used in the

documentation to unambiguously identify the parts that are required during each step. The part numbers also allow us to unambiguate the orientation of a part: part should always be assembled so that the engraved text is facing outward. This way, mirror assembly situations are avoided.

### 3.3.4 ROBOT-ASSISTED THERAPY

As described at the start of section 3.3, the Ono platform was originally conceived as a research tool aimed at robot-assisted therapy for children with Autism Spectrum Disorder (ASD). The term ASD encompasses a group of mental conditions characterized by social and communication deficits, fixated or repetitive behaviors, and sensory issues (American Psychiatric Association, 2013). The word “spectrum” is used to indicate that the disorder covers a wide range of severity, spanning from high functioning individuals that integrate well into society to severely autistic individuals requiring constant care. Studies suggest the frequency of ASD to be at 1 per 110 individuals (Baird et al., 2006).

Early intervention is a key component of treatment, and can greatly improve the sociability, functional independence, and quality of life of individuals affected by ASD (Rogers, 1996). Tools such as toys and stuffed animals are often used in the therapy of young children with autism as a way to increase motivation and attention. The use of animals in therapy is also a recurring theme (Nimer and Lundahl, 2007), though Animal-Assisted Therapy (RAT) suffers from a number of drawbacks. These drawbacks include the expense of training and caring for animals, their safety (both behavior as well as diseases and allergies), and their unpredictability. For these reasons, social robots have been suggested as an alternative means to conduct this type of therapy, leading to the concept of RAT. Prior work has already investigated the effects of robots in the socialization of children with ASD (Cabibihan et al., 2013; Kozima, Nakagawa, et al., 2005; Robins et al., 2005; Vanderborght et al., 2012).

In this domain, we identify a number of reasons why the use of a low-cost, open source social robot could be advantageous:

- Current studies in RAT specifically, and HRI in general are often limited by the availability of robots, leading to experiments with a limited number of participants. Access to small, inexpensive, and user-friendly robots would enable researchers to perform large-scale quantitative studies.
- Current ASD treatment therapy is time-consuming and expensive, and this problem is compounded by the limited availability of qualified therapists. While we believe it to be ill-advised to try to replace therapists with robots, we do think the technology has much to offer as a tool for therapists to use during sessions, and would help to streamline the therapy process. However, the availability of low-cost robotic platforms is a prerequisite to this.
- Continuing this thought, inexpensive social robots could also serve as a homework tool for children with autism. Children could learn new materials during therapy



sessions, and subsequently practice these lessons at home with aid of the robot. As an additional advantage, a scenario where parents help with the “homework” of their child can also serve to strengthen the bond between parent and child. Presently, a reoccurring problem is that severely autistic children can be much more sociable toward their therapist than toward their parents because child and therapist spend much more time together practicing social interactions.

- ASD covers a wide spectrum of conditions, and patients often require therapy that is personalized to their specific needs and to the extent of their ability. The open source paradigm can serve to address this need of adaptation and customization of the robotic platform toward the needs of a specific patient or subcategory of patients.

Continuing toward the exploration of this theme, we have performed two experiments within the context of RAT for the treatment of ASD. The setup and results of these experiments are detailed in the two subsections below.

#### 3.3.4.1 PILOT STUDY IN ROMANIA<sup>7</sup>



**Fig. 3.29** Child interacting with Ono during the pilot study

<sup>7</sup>This study was performed in collaboration with Cristina Costescu and Mihaela Vornicu from the Babes-Bolyai University, department of Clinical Psychology and Psychotherapy

A pilot study was performed to investigate the emotional response of children with autism toward Ono. The study was performed in Romania, where the robot was tested with five children diagnosed with ASD, aged 3 to 10 years old. Ono v1.2 was used during the study. The robot was controlled directly by the therapist performing the experiment. The robot functioned as an output device only, there were no integrated sensors in this prototype. In this “Wizard of Oz” setup, the therapist controlled the robot’s emotions directly via a joystick interface, thus emulating autonomous robot behavior. The test consisted of three phases:

1. The study began with an exploration phase. The children were given time to freely interact with Ono. Various aspects of interaction, such as facial expression mimicking, were observed.
2. In the second phase, the children were asked to identify the emotion expressed by the robot. The emotions happiness, anger, sadness and surprise were shown in random order, and each emotion was shown four times. The children were asked to pick the right emotion from the list of four possible answers.
3. Finally, the sessions concluded with the opportunity for additional play time.

Interactions were captured on video ( $M_8$ ) and were analysed post-hoc. The results of this pilot study are summarized in the tables below. Table 3.6 shows the interaction rates during the study. Table 3.7 shows the recognition rates of the four selected emotions. For each of the emotions, the table indicates the number of times the emotion was correctly identified, the number of times an incorrect answer was given, and the number of times children indicated they did not know the answer. For the recognition rates, only 16 measurements were obtained (as opposed to 20), as one child refused to participate in this part of the study.

Overall, these tests suggest that Ono has an overall inviting appearance that elicits interaction, though there are still several issues that need to be addressed. The participating children could easily identify happiness and sadness, but anger was frequently confused with being scared or sad, and surprise was often confused with happiness or sadness. We believe that major improvements can be made in this area simply by adjusting the DOF values for these emotions in software. On the other hand, the low recognition rates for these emotions may also be explained by the fact that these tests were performed with children with ASD, as opposed to normally developing children. A limitation of this study is the fact that there was no control group of normally developing children. Consequently, it is difficult discern whether incorrect responses were due to limitations of the robot, due to the emotional development of the children, or due to a combination of both factors.

On the technical side, we found the simplicity of the control setup to be beneficial. The robot could be used immediately by the therapists, without explicit training of the operator and without the aid of technical support personnel. This was essential, as we could not travel abroad to be present at the experiments.

However, many aspects of the control box interface proved to be suboptimal. Usage of the

joystick can be distracting for the children, and the short cable means that the whole setup can be unwieldy at times. In addition, the joystick control has an important downside. While it offers a quick method to change the robot’s emotions, it does not offer precise control, making it difficult to repeatedly select the *exact* same emotion. Finally, the addition of idle animations would be helpful to make the robot seem more lifelike when the operator is not actively controlling it.

During the free play phase of the study, most children continued to show interest in the robot. One child played a musical instrument to the robot, and another child tried to feed Ono. A third child asked to play with Ono after the study ended; and even controlled the robot himself using the joystick interface. The idea of making the robot controllable by children was unexpected. We found the idea interesting and chose to further explore this concept.

	Imitation <sup>a</sup>	Touching <sup>b</sup>	Verbal Initiation <sup>c</sup>	Engagement <sup>d</sup>
Child I	11	70	25	3:08 / 4:16 (73 %)
Child II	0	4	15	2:10 / 3:46 (58 %)
Child III	0	9	5	2:43 / 2:50 (96 %)
Child IV	7	13	3	2:09 / 3:29 (62 %)
Child V	0	0	40	0:23 / 2:32 (13 %)

<sup>a</sup> Number of times that the child had the same facial expression as Ono  
<sup>b</sup> Number of times that the child touched the robot  
<sup>c</sup> Number of times the child talks to the robot.  
<sup>d</sup> Time spent being attentive vs. total engagement time. (m:ss)

Table 3.6 Interaction rates

	Correctly identified	Incorrectly identified	Don't know
Happiness	15	1	0
Anger	3	10	3
Sadness	15	1	0
Surprise	6	7	3

Table 3.7 Emotion recognition rates

3.3.4.2 CHILD CONTROLLER<sup>8</sup>

The pilot study in Romania revealed an unexpected mode of interaction where the children control the robot directly, in conjunction with the therapist. This is in contrast to the Wizard of Oz-style RAT scenarios we had originally envisioned, where the controller would only be used by the therapist, and would be somewhat hidden from child’s view.

<sup>8</sup>The work described in this section was done by Pieterjan Mollé as part of his master’s thesis.



**Fig. 3.30** Wizard of Oz setup to test the child controller concept.

The concept of an emotion controller for children is an interesting idea, as it allows children to go one step further in interacting with the robot's emotions, allowing the child to experiment with emotions on a higher level of abstraction. For instance, starting with a negative or incorrect roleplay scenario, the child could iteratively tweak the social interaction scenario until a desirable result is achieved, learning more about that type of social interaction in the process. Another potential application of the child controller is to use it as a tool to help a child express their own emotions.

Some of the early concepts for the child controller included devices shaped like game console controllers, tablet-based interfaces, and even interfaces based on hand gestures. Eventually, it was decided to base the interface around a set of interactive playing cards, which could be used to change the robot's emotions or play back certain animations.

The concept is inspired by "*een doos vol gevoelens*" (translated: a box full of emotions), a therapy tool currently in use at the therapy center where the tests were done. The card-tool is aimed at children between ages 4-7, and deals with the basic emotions happiness, sadness, anger and fear.

In our scenario, we have created new emotion cards that can be detected by the robot through computer vision. When a child shows the robot a specific card, the robot would subsequently portray the emotion that is associated with that card. Eight different cards were designed, covering four emotions (happy, sad, angry, fear) and two degrees of intensity (mild and intense).

The concept was evaluated through a Wizard of Oz experiment involving four children with autism, with ages ranging from 5 to 10 years old. The setup of the experiment is shown in figure 3.30. The children were tested individually, and each session lasted

approximately one hour. The sessions were captured on video ( $M_8$ ). The sessions were structured similarly to the regular sessions, though with the presence of the robot as an extension to the “doos vol gevoelens” tool.

The sessions were done individually, and were mediated by a therapist. Four therapists were involved in the study, as each child has their own therapist. The robot was put on a table, with the emotion cards placed in front of it. A backdrop decor was placed behind the robot. This backdrop serves two purposes: (1) the backdrop hides the robot operator, who is seated behind the it, (2) the backdrop contains a camera through which the operator can see. The operator is responsible for manually changing the emotion of the robot in response of the child showing an emotion card. The entire session is captured through two cameras, one hidden in the backdrop and a second one positioned behind the child.

The concept was met by positive reactions from both the therapists as well as the children. Preliminary results suggest the system is intuitive, the users were comfortable with the system after experimenting with it for a short amount of time. In a follow-up interview, the therapists indicated that they thought that the presence of the robot enhanced the motivation and attention of the children during the session. They also liked the background decor, even though it was originally intended as a solution to hide the operator in the Wizard of Oz setup.

A number of shortcomings were also identified during the study. To begin, the cards should have a better graphical design and should be made from a more durable material. In addition to the emotion cards, the system should also incorporate situation cards that cause the robot to play a short scenario. Finally, a number of animations should be added in order to attain a better illusion of life. These animations should include a wake-up/sleep behavior, as well as simple idle animations, such as blinking and sleeping.

### 3.3.5 SUMMARY

This section discussed the first generation iterations of the Ono social DIY robot. We described the original inception of the project, building upon previous work with the social robot Probo and state of the art in social robots for use in therapy. New insights and the combination with the maker movement paradigm defined the first concept. With digital fabrication and the social context in mind, this first generation created a believable social character named Ono ( $G_5$ ) that has a strong focus on emotions via facial expressions ( $G_3$ ,  $G_4$ ). The experiments evaluated these emotional expressions of the robot by the end-users in the context of RAT for children with ASD. While the digital fabrication and DIY design allowed for an open ( $G_1$ ), easy to build ( $G_2$ ) and low-cost ( $G_9$ ) solution, the experiments gave new insights to improve instructions and parts in order to better guide the flow ( $G_6$ ) of the assembly process for the second generation.

## 3.4 ONO – GENERATION 2

The second generation of the Ono robot is marked by a complete redesign of the entire robot. The visual appearance of the robot is the only constant between the two generations, as all internal aspects were changed. The experiments done with the first generation prototypes yielded valuable insight into aspects such as manufacturing techniques, reproduction, and robot usage. However, the prototypes did not incorporate all elements required for a true robotic system. Consequently, the first generation Onos functioned more or less as high-tech hand puppets.

In late 2014, the Ono robot was completely redesigned. The second version incorporates lessons learned from our experiments, and expands the robot's functionality resulting in a more capable robotic platform that is viable for use in HRI experiments. There are a number of factors that influenced our decision to to a near-complete redesign of the robot's internals, as opposed to carrying out incremental improvements to the existing design. First of all, we decided to include low-cost FDM 3D printing (i.e. RepRap and derivatives) as a second production technique. As a manufacturing technique, 3D printing allows us to simplify many small and intricate parts within the design of certain modules, leading to mechanisms that perform better and are easier to assemble. As a second factor, we decided to use a new interface standard for the modules based on redesigned snap connectors. Consequently, all modules were changed in one way or another, though the new eye module functions on a fundamentally different principle, necessitating a complete module redesign.

The changes to the module interface standard meant that changes to the skeletal frame structure were inevitable. We took this as an opportunity to employ a new design strategy, which leads to a stronger, more rigid frame. Additionally, the second generation frame has recesses to accommodate the robot's power supply and control electronics, meaning that the first generation control box can be eliminated.

In the redesign process, we decided to migrate our design from Siemens NX to Solidworks, a different parametric CAD software suite. In our experience, there are a number of small differences between the two that make Solidworks better suited for designs that involve large amounts of laser-cut components. For instance, the new software allows us to save certain sketch geometry, such as the geometry of a snap connector, as a reusable sketch block. We rely extensively on this functionality because it speeds up the skeleton design process immensely. As an added benefit, Solidworks is also much less expensive and more popular than NX, lowering the barrier to modification a bit. It is also frequently used as a design tool in complex open source hardware projects, especially those that originated from an academic environment. Some examples include OpenHand (Ma et al., 2013) and WoodenHaptics (Forsslund et al., 2015).

Finally, in the redesign, the electronics and software stack were updated to lay the foundations for a system that is suited for human-robot interaction scenarios. The electronics and software stack of generation one can be described as minimal, it was intended primarily to test the mechanical design and was not well suited for interactive scenarios, as

illustrated by the RAT experiments. The upgraded setup is built around the Raspberry Pi Single-board Computer (SBC), and allows the operator to control the robot over WiFi through a web browser-based interface that offers multiple, task-specific applications.

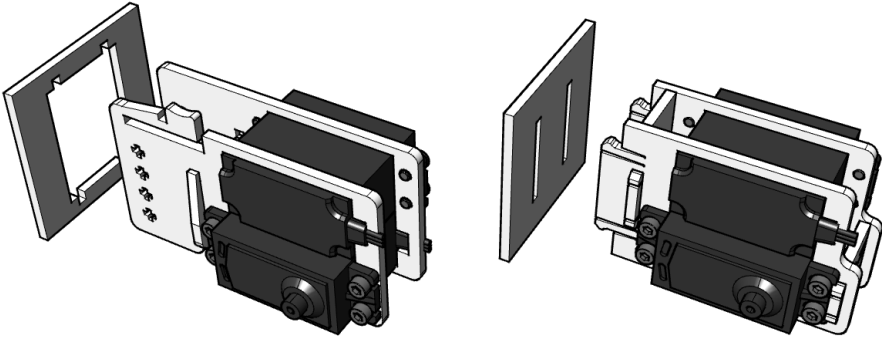
Within the second generation of the Ono robot, we distinguish the following versions:

- **ONO V2.0** – The initial prototype of this generation, designed for use during the assembly workshop at UNN (section 3.4.2). Key differences include an updated module interface standard, a redesigned eye mechanism, an updated skeleton design, and a new SBC-based electronics architecture.
- **ONO V2.1** – The biggest improvements though, are in the software and electronics of the robot. After using a perfboard-based circuit board in v2.0, we designed a custom PCB to interface the Raspberry Pi with the robot's sensors and actuators. The first version of the app-based web interface was also introduced in Ono v2.1.
- **ONO V2.2** – This version comprises the changes and improvements made ahead of the workshop at the HRI summer school (section 3.4.3). The most important change is the redesigned electronics board, leading to better reliability, new sensor inputs, and an audio speaker output. On the mechanical side, this version includes minor tweaks to the eye design, making the eye module more durable and easier to assemble. A second mechanical improvement includes a redesigned bracket for the robot's logic power supply and ethernet connector. Finally, the new foam parts have precut stitch holes, making the foam skin assembly easier and quicker.
- **ONO V2.3** – The main changes in this version are software improvements. This includes changes to the core software architecture, as well as the addition of new apps, such as the Social Script app. Additionally, capacitive touch sensors were integrated into the hands, feet, and head of the robot.

### 3.4.1 SKELETON AND MODULE IMPROVEMENTS

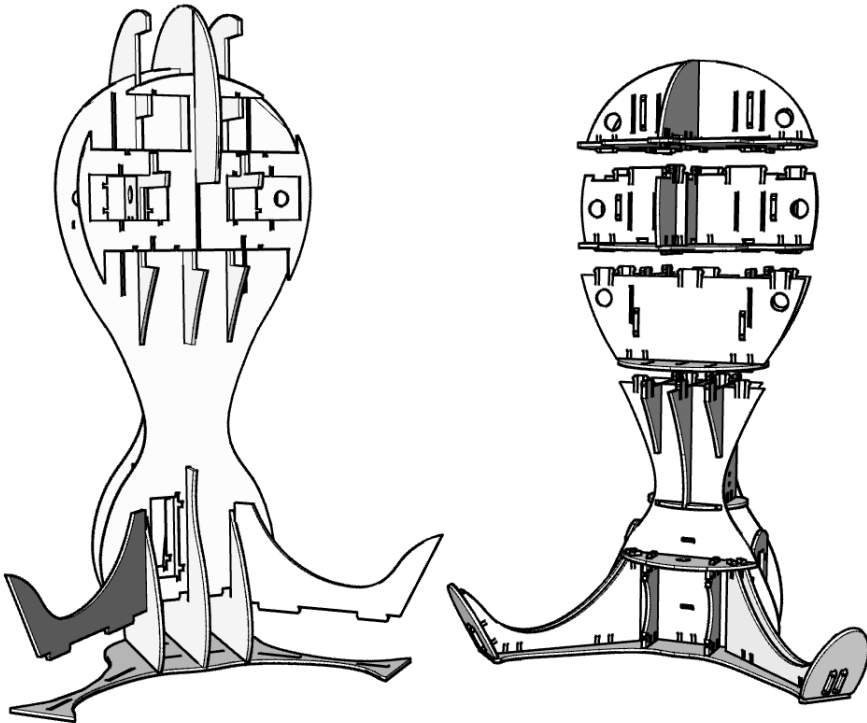
From generation two onward, a new mechanical interface standard was chosen to connect the modules to the frame of the Ono. Figure 3.31 shows a comparison between the old and new module standards. The first generation system relies on a single lever arm to lock a module in place. To remove the module, this lever needs to be depressed by the user. However, this proved suboptimal as the levers tend to be hard to reach once the modules are installed in the frame. Because only a single cantilever snap is responsible for holding the module in place, the system does experience some play in the vertical direction. The third problem with the old module interface is that the geometry is relatively large, taking up valuable space inside the volume of the robot.

The module interface of generation two is inspired by the updated skeleton snaps, described in section 3.3 and shown in figure 3.20. The interface consists of two double-sided cantilever snaps, resulting in a higher retention force. The lever system from the first generation was eliminated. Additionally, the geometry of the snap itself was adjusted so that



**Fig. 3.31** Comparison of the first generation module interface (left) vs. second generation module interface (right)

the connection can be undone. The mating geometry was standardized to two parallel slots of  $3 \times 30$  mm, spaced 18 mm apart.



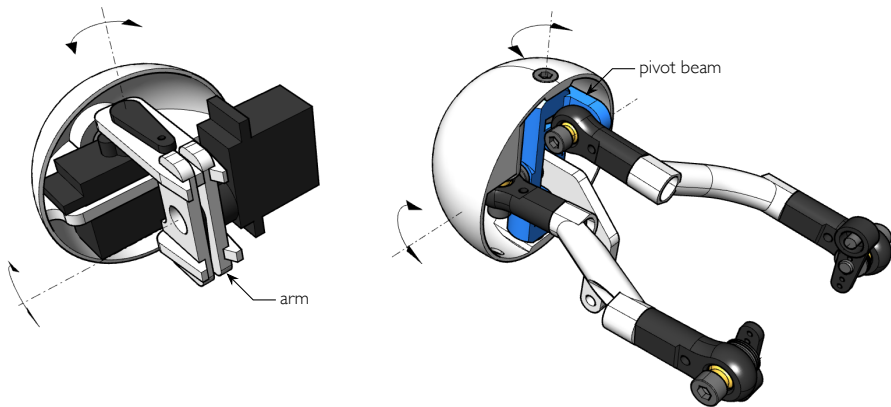
**Fig. 3.32** Comparison of the first generation frame (left) vs. second generation frame (right) construction



The updated module standards necessitated modifications to the design of the skeleton to make the new modules fit. Instead of only updating the mounting locations, we chose to use as an opportunity to completely upgrade the design of the skeletal frame. Figure 3.32 shows a comparison between the old (left) and the new (right) skeleton designs. Both frames are partially exploded in order to highlight the differences in the design approach.

In the old design, the shape of the robot was sliced vertically in two orthogonal directions, parallel to the front plane and parallel to the side plane. In each slice, slots are added to allow the slices to intersect. However, because of the size of the slices and the length of the slots, the laser-cut parts lose a fair amount of rigidity.

Even though the same material and the same production technique are used in the re-designed frame, the new design approach leads to a stronger, more rigid frame. The new frame is composed of smaller interlocking sub-units. These sub-units flex less because they snap together, leading to better rigidity. Additionally, the laser-cut parts are smaller and can be nested more efficiently, leading to less waste and better material use. This way, the amount of sheet material used was reduced by a third.



**Fig. 3.33** Mechanism of the eye module. Old direct-drive mechanism (left) vs. new linkage-driven mechanism (right).

The third major change in the second generation of Ono is an upgrade of the mechanism responsible for the motion of the eyeball, shown in figure 3.33. The first-generation of eye modules were built using solely laser-cut parts and standard parts. Movement of the eyeball was achieved using a simple serial kinematic chain: the eyeball itself is attached to a first servo, which is responsible for the horizontal (pan) motion. This servo is in turn connected via an arm to a second servo, responsible for the vertical (tilt) motion. The small components inside the eye, such as those of the connecting arm, are not very well suited to be made with a laser cutter. Furthermore, the centers of rotation of the two DOFs do not intersect, meaning that the movement is not entirely concentric.

The second-generation eye module design relies on a different principle, made possible

by including DIY 3D printing as the second production technique. The eyeball itself is suspended in a two-axis gimbal system, suspended by a single 3D printed component dubbed the *pivot beam*. The eyeball is actuated by two ball-joint linkages, with each linkage responsible for the motion of one principal axis of rotation. The resulting system is more robust, rotates concentrically, and exhibits less jitter and backlash.

### 3.4.2 WORKSHOP AT UNN



**Fig. 3.34** Participants of the workshop posing around the robot they built in one day

The design of the second generation of Ono started in the summer of 2014, with the goal of using the new design for our workshop at the Lobachevsky State University of Nizhny Novgorod. The workshop was part of a summer school organized by UNN's department of social sciences. Consequently, the workshop served as the first experiment of our newly redesigned robot. During this one-day workshop, 15 participants worked to assemble one Ono robot from parts we brought. Participants were university-level students coming from various disciplines within social sciences. The participants had little or no experience in constructing physical objects.

As part of the workshop preparations, we brought basic tools (e.g. screwdrivers and pliers), components bundled per subassembly, an assembled set of electronics, and a preassembled robot. The preassembled robot served as a reference during the build process. Participants were also provided with detailed handouts with assembly notes and figures, though we found the most effective technique to teach the assembly process was to let participants copy the robot based on the example. The one-day workshop was divided into the following phases:

1. The workshop started with a short introductory presentation to give an overview

of the planning, to give some background information on the Ono project, and to pass along general practical assembly tips.

2. In the first phase of the workshop, the five required modules were assembled. The participants organized themselves into smaller groups, with each group in charge of one particular module. This is possible because the modules serve as independent subassemblies, which can be assembled separate from the rest of the robot.
3. In the second phase, the skeletal frame was assembled. This was done by the participants that assembled the “simpler” modules. The groups building the eye modules required more time due to the higher number of components.
4. After finishing the modules and the skeleton, the modules and electronics were mounted onto the frame. This step also included some light soldering work in order to attach the power supply to the power cord.
5. In the final phase, the foam and textile skin was attached to the robot. The robot was then started, the servos were calibrated, and a demo script was run to test the completed system.

As an exploratory study, this workshop yielded a number of helpful insights. The modular design of the robot proved advantageous during the workshop because it stimulates parallelization: participants could easily organize themselves into groups focused on building one specific sub-assembly. Naturally, there are limits to this approach: at a certain point in time, subassemblies need to be joined together, at which point the groups are forced to converge.

Participants also expressed a sense of fulfillment at the end of the workshop. Most of them had little or no experience in making things, and found that during the workshop, they were exposed to many different techniques for making. One participant in particular described the workshop as an eye-opening introduction into Maker culture, and decided to do his master’s thesis on DIY social robots for therapy.

The workshop also revealed a number of minor shortcomings in the design of the robot. Generally speaking, these shortcomings mainly relate to parts that are too fragile or too difficult to assemble. One participant regretted that the workshop was primarily about assembling the robot, and would have preferred that a portion of the workshop be dedicated to summarizing the major aspects of the software of the robot.

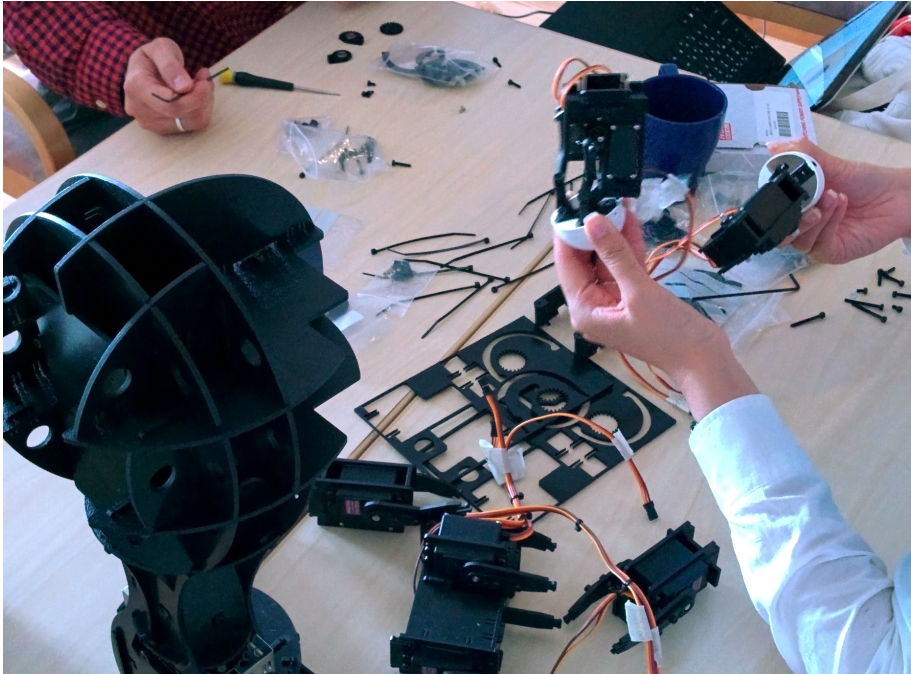
As a final remark, we also found that the participants – students and faculty members – were very much open to the idea of building/modifying their own tools for use in therapy. In our experience, this is not always the case with practicing therapists.

### 3.4.3 WORKSHOP AT HRI-SS

The second assembly workshop was organized during the 2015 summer school on human-robot interaction in Åland, Finland. This workshop was larger in scope and scale than the

workshop at UNN, and builds upon improvements prompted by the previous workshop. The workshop took place in three consecutive afternoon sessions of four hours. Roughly 20 HRI researchers participated in the workshop, and together they built six Ono robots. We asked participants to come only if they could attend all three sessions, as each session builds upon the previous one. Sadly, some participants could only attend one or two sessions due to reasons such as travel arrangements or overlapping workshops. The three workshop sessions were organized as follows:

- *Session 1* – In the first session, the modules and the skeleton of the robot were assembled.
- *Session 2* – In the second session, the modules and the electronics are connected to the skeleton frame, electrical wiring is done, and the skin is attached.
- *Session 3* – In the last session, the skinning is completed and small scenarios are programmed using the API.



**Fig. 3.35** Workshop participants assembling an Ono.

Documentation for the workshop was provided through a wiki with step-by-step assembly instructions with photos. We had intended to give participants direct access to the wiki so that they could edit and improve instructions if necessary. This approach was inspired by work of Schelly et al. (2015). Unfortunately, because of the unstable internet connection at the venue, we were forced to fall back on printed handouts of the wiki.

This workshop expands upon the scope of the previous workshop (section 3.4.2) by including programming and soldering activities. For the third session, we also brought extra sensors and materials in order to enhance the programmed robot scenarios with custom tangible interfaces.

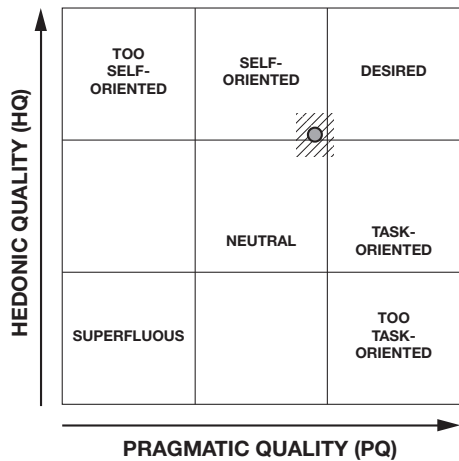
At the end of the workshop, participants were asked to fill in a survey. The survey consisted of the following questions:

- General personalia questions: name, age, gender.
- 7-point Likert scale statements ( $M_4$ ), with a value of one indicating complete disagreement, and seven indicating complete agreement:
  - “The workshop has given me more insights in the design of social robots.”
  - “The workshop has given me more insights in the construction of social robots.”
  - “I am interested in using the Opsoro system to develop my own robot..”
  - “I am interested in using the Opsoro system to conduct new HRI experiments.”
  - “I am interested in using Opsoro robots such as Ono to develop new software applications.”
- The AttrakDiff questionnaire ( $M_2$ ).
- Open questions ( $M_6$ ):
  - “What aspects did you really like during the workshop?”
  - “What was the most annoying aspect, or where did you experience the most problems?”
  - “What did you learn from the workshop?”
  - “How could we make the workshop better in the future?”
  - “What is still missing in the Opsoro system?”
  - “What is still missing in the Ono robot?”
- The UX Curve ( $M_5$ ).

Overall, the workshop was a success in our opinion. All groups were successful in building the robot, and feedback from participants shows a generally positive trend. In general, participants noted that they liked how the workshop taught a variety of practical skills, such as soldering, in a short amount of time and applied to a realistic project. As with previous workshops, most problems reported by participants were related to technical issues and software bugs. For instance, one robot did not work because there was a problem with one of the custom PCBs. Another problem that was frequently reported by participants was the lack of tools. As the workshop took place on at a remote location, we were limited in the amount of tools and materials we could bring. During the workshop, this caused a bottleneck as people were taking turns using a specific tool. Finally, we note that the

software API is still too complex for casual use. Most groups used their time in session three to finish assembling and configuring their robot. Only one group created a custom scenario and interface using a Makey Makey board.

In total, 14 participants completed the survey. The average age of respondents was 27.6, with a spread of  $\sigma = 2.65$ . Ages ranged from 24 to 34. Gender distribution was roughly equal, with eight male and six female respondents. Results from the AttrakDiff questionnaire are shown in figure 3.36. The overall product attractiveness (*ATT*) is rated at 1.65. The results position the system just outside the quadrant of desired products. The AttrakDiff score could be improved in part by addressing the technical issues we experienced. Doing so would improve the system’s pragmatic quality (*PQ*). The word pairs *technical – human* and *cheap – premium* scored the lowest, indicating that these product aspects should also be improved.



**Fig. 3.36** Results of the AttrakDiff questionnaire. The hatched area represents the confidence rectangle.  $n = 14$ ,  $PQ = 0.84 \pm 0.29$ ,  $HQ = 1.09 \pm 0.32$

Results from the Likert scale questions, summarized in figure 3.37, show a generally positive trend. The results hint that respondents were more interested in the techniques and methods shown in the workshop rather than the Opsoro system itself. This sentiment is mirrored in the responses to the open questions. When asked what they learned from the this workshop, one respondent answered: *“I had many insights on technologies available to build a social robot, and I really appreciated the hand work part.”* Another participant summarized the learning experiences as: *“Concepts of modular robotics. Aspects of robot design. A lot about assembling laser-cut + 3D printed parts.”*

Finally, results of the UX curve tool are shown in figure 3.38. Again, the curves show an overall positive trend. Still, the tool revealed several points of difficulty or frustration. One respondent remarked that time was short for the amount of work that needed to be done, and that time pressure was a cause of stress near the end of each session. Rightfully, they remarked that it would have been better to construct a less complex robot (i.e. less DOFs)

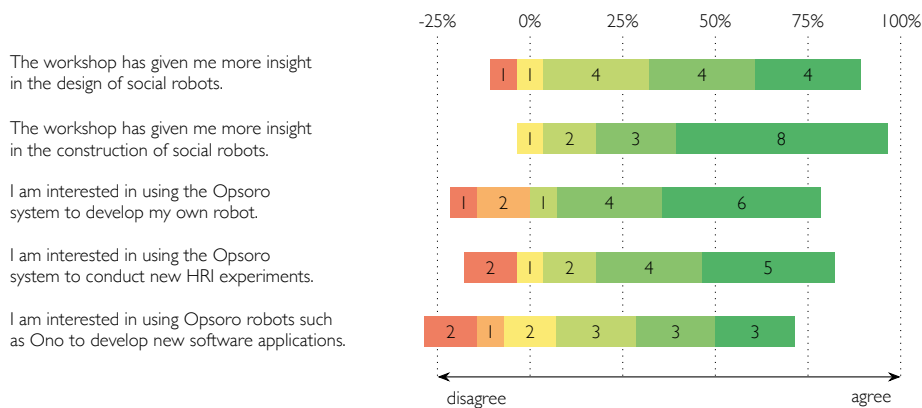


Fig. 3.37 Results from the Likert scale questions.  $n = 14$ .

during the workshop. Another respondent remarked that sewing the foam together is a difficult and time-consuming part of the assembly process. Finally, a respondent remarked that they started the workshop with a neutral disposition as they did not know what to expect. They note that their excitement grew as time progressed, and that seeing the robot work for the first time was a very satisfying moment.

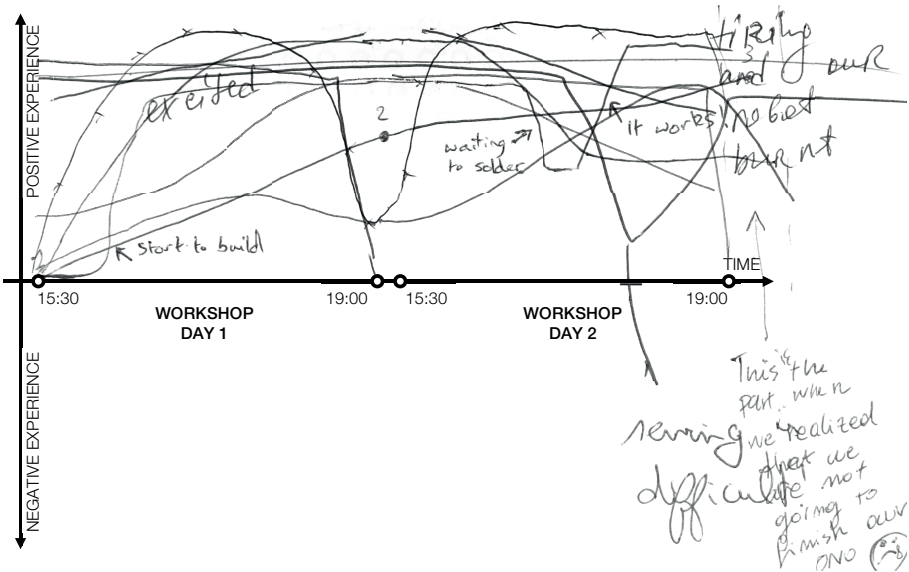


Fig. 3.38 UX curves drawn by the the participants of the workshop.  $n = 10$ .

### 3.4.4 USING ONO IN THERAPY<sup>9</sup>

As a continuation of the work described in section 3.3.4.2, the newest version of the Ono robot was re-evaluated for use in therapeutic applications. During these tests, two separate therapeutic contexts were explored:

- *Context 1* – An independent psychotherapist located in Kortrijk, Belgium. The experiments done within this context involved one specific patient. The patient is 15 years old and is diagnosed with attention deficit hyperactivity disorder (ADHD).
- *Context 2* – A therapy and day care center in Deinze, Belgium. The center is primarily focused on children aged 4-12 years old with less severe forms of ASD.

Initial interviews ( $M_7$ ) with therapists from both locations affirmed the importance of being able to create custom social scenarios for therapeutic robots. This context is another example of a situation where von Hippel's idea of *sticky information* comes into play (Hippel and Katz, 2002). The therapists know perfectly well what is important for a social scenario, and how a scenario should be shaped for specific patients. Roboticians have only limited access to this tacit knowledge, hence why it makes sense to create tools for therapists instead of creating scenarios directly.

The interviews, as well as tests with children confirmed that the size and shape of the robot was appropriate for various types of therapies. Additionally, the therapists of context 2 revealed an unexpected benefit of the current embodiment design of the robot. The robot has a fairly generic, nondescript shape. However, the visual character of the robot can easily be customized using old children's clothes and costumes. This is an important benefit for use in social scenarios: different archetypes – such as hero or villain – can be created quickly.

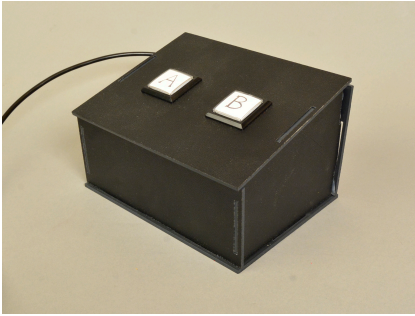
As suspected, the interviews with therapists confirmed the importance of allowing therapists to program custom scenarios for the robot. As such, the co-design process described in this section was primarily focused on designing a new software interface to create custom scenarios for therapy. As a first step, the existing Blockly-based visual programming interface was evaluated. This is the same programming interface that was used during our workshop at the HRI summer school (section 3.4.3).

To evaluate the Blockly interface, a small scenario was designed and programmed for the patient of context 1 in conjunction with his therapist. The scenario consists of a short introduction of the robot character, followed by a quiz with questions on moral topics such as bullying and drug use. Because Ono currently does not have voice recognition software, a custom controller was created as a quiz interface. The quiz controller, shown in figure 3.39, has two physical buttons that allow the user to answer questions.

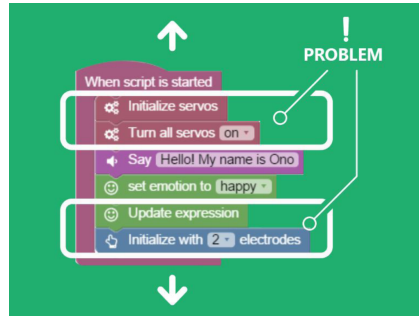
The test with the patient was very successful, more so than expected. Initially, we feared

<sup>9</sup>The work described in this section was done by Marie Van den Broeck as part of her master's thesis (2016).





**Fig. 3.39** Quiz control interface



**Fig. 3.40** Blockly API issues

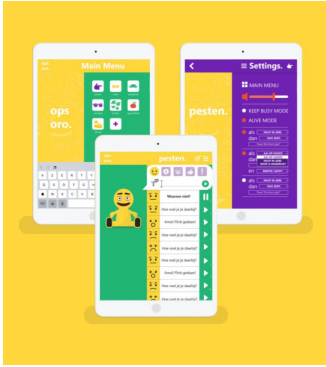
that the patient would not be receptive to the concept and the character of the robot because its appearance was designed for a different user group (see section 3.3.1). In contrast with our expectations, the patient indicated he enjoyed the experience, and wanted to try the scenario multiple times. The robot also introduced a roleplaying element into the session, which functioned as starting point for deeper conversations.

The technical aspect of the experiment was less successful, as the test revealed that the Blockly API is much too complex and verbose for casual use, reaffirming one of the conclusions of section 3.4.3. The scenario was comparatively simple; the quiz was comprised of only five questions. However, the corresponding script was very large and unwieldy, encompassing multiple screens. For this usage scenario, the design of the Blockly API is too verbose and low-level. This problem is illustrated in figure 3.40, showing how even simple scripts require superfluous commands to function.

Further interviews and brainstorming sessions with the therapists from both contexts revealed a number of insights. The interviews revealed that the therapy app should offer a simple way to create Wizard-of-Oz dialogs. The use of actual programming constructs (e.g. variables, functions) is not required nor desired. The scenario interface should be easy to learn, favoring simplicity over functionality. The final concept of the app is essentially a remote control interface that allows therapists to prepare and play back lines of dialog. Each line of dialog is comprised of a facial expression combined with a line of text or a sound file. Dialogs can be prepared on beforehand and saved to a file, allowing therapists to create different sets of dialog lines for different scenarios or different patients.

After the concept requirements were finalized in dialogue with the therapists, an initial mockup of the interface was created. The mockup is made up of images of the most important interface screens (fig. 3.41). A tablet was chosen as the primary device to test the mockup because tablets are already used in therapy sessions. The mockup images were brought to life using InVision, an application that lets interface designers quickly prototype new interfaces by adding basic interactivity to mockup images.

The InVision mockups were tested with the therapists, which led us to discover two important shortcomings. First of all, the therapists emphasized the importance of file man-



**Fig. 3.41** Interface mockup



**Fig. 3.42** Social Script app in use

agement tools, as they have to deal with multiple patients and many different scenarios. The second insight is that there needs to be a direct mode; a way to create and play a new line of dialog while a therapy session is in progress. This is important in order to reply to unexpected responses from the patient. These improvements were included in the implementation of the Social Script app, shown in figure 3.42. As a final remark, we notice a parallel between the design of this software interface, and the design of a modular toolkit. As it turns out, a very flexible interface paradigm was not ideal, as it is much harder to learn than the more prescriptive interface of the Social Script app. A similar design dimension is found in the design of the hardware modules. As argued in section 1.1.5, a toolkit designer must strike a balance between flexibility and ease of use.

### 3.4.5 SUMMARY

The second version of Ono represents a complete redesign, based upon the lessons learned from the first generation of Ono designs. The first point of difference is that the new version incorporates a second production technique: low-end FDM 3D printing. This change affects the reproducibility ( $G_1$ ) of the system, as the design now requires extra machines to manufacture. In our opinion, this change is justified because it makes the design much easier to assemble ( $G_2$ ) and the increased performance of the modules improves the emotional expressiveness of the robot ( $G_3$ ,  $G_4$ ). Furthermore, low-end 3D printing is one of the staples of contemporary DIY culture, and the technique is readily available in FabLabs, through online services, or through homemade machines.

The experiments described in this section delved much deeper into the assembly process, exploring the difficulty level ( $G_2$ ) and investigating problems that hamper the state of flow ( $G_6$ ). The second version of Ono was also tested in a therapeutic context, affirming the appropriateness of the Ono character ( $G_5$ ) outside of the context of ASD therapy.

But the biggest change in this generation is the new electronics architecture, based upon the open Raspberry Pi platform that made the web-based user interface possible ( $G_{10}$ ).

Whereas the first version of Ono could be described as an animatronic puppet, the re-designed version is a fully functional robotic system, capable of (semi-)autonomous behavior. In contrast to other social robots, the robot now serves as a web-platform offering different applications to the users from very easy Social Scripts (1), over to visual programming with Blockly (2), to basic scripting with Lua (3), and to full programming with Python (4). Already with these four steps a wide range of programming skill-sets can be addressed from novices to real software developers. This approach fits with the concept of positive flow ( $G_6$ ) matching skills with challenges and draws similarities with the DIY hardware, although the hardware still lacks easy customizability of user's own character design. This is further addressed in the next section in which the Ono technology is transformed into the Opsoro platform, focusing on the need to design custom social robots.

### 3.5 OPSORO PLATFORM

One of the premises in this project is that DIY adaptability can be advantageous in many applications of social robotics, and the merit of this idea was confirmed during our experiments and during interviews with therapists. Following the experiment at UNN, we decided to take this idea one step further by repurposing the components of Ono as craft materials for the construction DIY social robots.

The workshop at the TEI conference marked the first time we used the Ono modules as part of a design toolkit to build new social robots. Based on the success of this workshop, we chose to focus our efforts on expanding the functionality of our technology to support and facilitate the design of custom social robots. To this end, we coined the term Opsoro – *Open Platform for Social Robots* – as the name for the underlying technologies upon which Ono is built. The focus of the Opsoro platform is to enable non-experts to go from a character concept to a functional social robot, emphasizing low-cost and DIY aspects and aiming primarily at characters with animated facial expressions and limited body/limb motion.

Following the conference workshop, we implemented the workshop's concept as a full-semester course assignment with industrial design students from our university. We have organized a course assignment on this subject, which took place in spring 2015. Internally, we adopted the term "*The illusion of life*" as an unofficial moniker for the course. The title references the seminal work on cartoon animation of the same name, written by two Disney master animators (Johnston and Thomas, 1995). The philosophy of the book is a good analogy for the goals of our work. We do not aim to build robots, we wish to create physically embodied characters. Just as animators use pen and paper to suggest that a drawing is alive, we aim to use robotic components as a creative medium to design characters.

Results from the workshops and interviews with potential end users prompted us to further simplify the toolkit. This simplified version is designed with a focus on applications in STEAM, as market research indicated that this target audience offers the best oppor-

tunities for commercialization. Consequently, our efforts were directed toward reducing cost and complexity of the system while safe-guarding the DIY and hackable nature of the preceding designs. In addition to the simplified modules, this version also incorporates a new connection principle to attach modules to the embodiment. Finally, the redesigned toolkit was tested with 48 secondary school students in late 2016.

- **OPSORO V1.0** – The initial version of the toolkit. Components from the toolkit were extracted from the design of the Ono robot, and were used during a one-day workshop at the TEI conference. Cardboard and craft materials were used to construct the embodiment.
- **OPSORO V1.1** – The version of the toolkit that was used during the Illusion of Life course assignment. It incorporates module improvement from Ono v2.1. Embodiments were constructed using digital manufacturing techniques, such as laser cutting and CNC milling.
- **OPSORO V1.2** – A simplified version of the toolkit aimed at secondary school STEM education. This version of the toolkit uses simplified modules and incorporates a new connection system: the Opsoro Grid.

#### 3.5.1 DESIGN WORKSHOP AT TEI



**Fig. 3.43** Participants working on the “Michael Jackson” robot.

To investigate the concept of using Ono components as the basis for a toolkit, we organized a one-day workshop during the 2015 conference on Tangible, Embodied and embedded Interaction (TEI) at the d.school in Stanford. During this workshop, participants were invited to create new animatronic creatures using modules from Ono in conjunction with traditional craft materials such as cardboard and foamcore.

The main aim of our workshop was to introduce participants to social robotics through a hands-on approach. We believe that the audience of the TEI conference is appropriate because there is a large overlap between social robotics and tangible interaction, and as such the topics covered in this studio should be of interest to the participants. On the other hand, the background of participants is sufficiently distinct so that it offers us new perspectives from outside the field of HRI. Furthermore, participants from the TEI audience are usually already familiar with some of the techniques and technologies that we use during the workshop. This is important due to the time constraint inherent to the workshop format.

The workshop's goal is to allow participants to design and prototype their own robotic creature. To this end, we provided a number of preassembled modules, a stand-alone electronics unit, and craft materials to build with. Four different kinds of modules were provided: mouths, eyebrows, eyes and joint modules. Craft materials included cardboard, glues, fabrics, and foam. We also provided custom plastic connectors to interface the modules with cardboard structures. Simple, low-threshold prototyping techniques, such as cardboard prototyping, were used throughout the session. This choice was made due to time and infrastructure constraints, though it also helped to underline the idea that an iterative process is important in the design of interactive systems. The workshop was divided into four phases:

1. *Concept Generation* – The workshop was started with a brief introduction into the field of social robotics, including a short discussion on its history as well as inspirational examples of HRI. This was followed by a demonstration of Ono and explanation of the design philosophy behind it. Finally, participants were introduced to our social robotics toolkit, going over the possibilities and limitations of the kit. The phase ended with a brainstorm. Participants were given time to come up with a concept and scenario for a new social robot.
2. *Embodiment* – In the second phase, participants were given the tools and materials to turn their chosen concept into a working prototype. We provided the toolkit, which contains modules for eyes, eyebrows, mouth and joints, as well as craft materials to create the embodiment of the robot. The craft materials we chose were selected because they allow quick design iterations with minimal effort.
3. *Creating Interaction* – After completing the embodiment of the robot, all modules are connected to the electronics in order to bring the creatures to life. The electronics consists of a Raspberry Pi with a custom daughterboard to control the servos. The electronics and the power supply were preassembled into stand-alone units so that participants did not have to do any electronics work outside of plugging in the

servos. In addition to the modules, which function as output devices, we also provided a number of Makey Makey boards (J. Silver et al., 2012). These boards can be used to turn any conductive surface into a touch sensor, giving the robot basic sensing capabilities. The robots can then be controlled via a web interface running on the Raspberry Pi. We created a visual programming language based on Blockly<sup>10</sup> to allow participants to program behaviors. Simple interactive scenarios can be created using this language by dragging and connecting puzzle-shaped blocks.

4. *Demos and Reflections* – After every team completed their animatronic prototype, the groups demonstrated their robot to their peers. A short movie of each human-robot interaction scenario was recorded. A short discussion was conducted to see what participants thought of during the design process and to see if this lo-fi prototyping approach could be useful in other researchers' work. Afterwards, participants were also asked to fill in a short survey.

In total, 6 people took part in the workshop. Participants teamed up in two groups of three, building two new robots. The first was a pirate character that would throw a tantrum when his rum bottle is taken away. The second character, shown in figure 3.43, was inspired by persona of Michael Jackson. It would perform a rendition of “Billy Jean” whilst moving its fedora hat. Unfortunately, two participants had to leave halfway through the workshop due to an overlapping session. This posed some difficulty as they were members of the same team, though the situation was addressed through help from the organisers.

At the end of the day, the participants were asked to fill in a short questionnaire. The questionnaire consisted of the following parts:

- General personalia questions.
- A user experience curve ( $M_5$ ).
- The AttrakDiff questionnaire ( $M_2$ ).
- Open questions ( $M_6$ ):
  - “What did you like about the workshop and the robot platform?”
  - “What did you dislike about it?”
  - “Are there any other suggestions or remarks?”

Figure 3.44 shows all the UX curves drawn by the participants and figure 3.45 summarizes the results of the AttrakDiff questionnaire. The responses to the open questions, as well as general information on the background of the participants are summarized in table 3.8.

Both teams were successful in designing and building a custom robot, though the workshop helped identify a number of weaknesses in the toolkit. Feedback from the questionnaire as well as personal observations during the workshop show that most problems are

<sup>10</sup>Blockly – <https://developers.google.com/blockly/>

	Gender	Age	UX curve notes	Liked	Disliked	Other remarks
SM	F	27	Great,examples on the slides. Interesting research projects you do. I'm more a,"compact info in slides" type so I especially liked the beginning.	Great,platform, good insights :). I personally like to learn more via slides. (=more info in less time).		
SK	F	26	When I,saw the movement making expression, it was the most exciting!	I'm,now working on a tangible and interactive product for the thesis project and,i found one of my insights and researchs that emotion makes fun somewhat is,in common in design field. And it's impressive that the robot makes many,expression than I expected.	I hope,the robot would be prettier. If you try to test it and to give for young,children, it should have more friendly looking. When I glanced at it it was a,little scary.	Nop!,It's excellent! Thank you :)
HF	F	25	Cool,demo. Sussing out details. Lunch/Partners leave. Partners left so I had to,try to do it on my own (time limit stress). Help from organizers	Fairly,easy to use with some experience. Open to do whatever so I didn't feel afraid,to mess up, which has held me back in the past.	Small,bugs – minor complaint, understandable given stage of development	Obvious,things like sussing out the bugs / adding save feature.
SB	M	25	Learned,a lot about Ono. It would've been nice if we programmed more. It is a great,platform to start robot prototyping.	It is a,very easy to use platform. Other than the bugs, every thing was fun to play,with.	Bugs,with the visual programmer.	It was,very open ended which made it fun. The context you laid out at the beginning of the workshop made it fun.
XZ	F	21		Brainstorming,proto!, team dynamic, very amazing vision!!, presenting/demoing it, sound!	Bugs,,felt like a guinea pig	
KG	M			<i>no data</i>		

Table 3.8 Results of the questionnaire's open questions

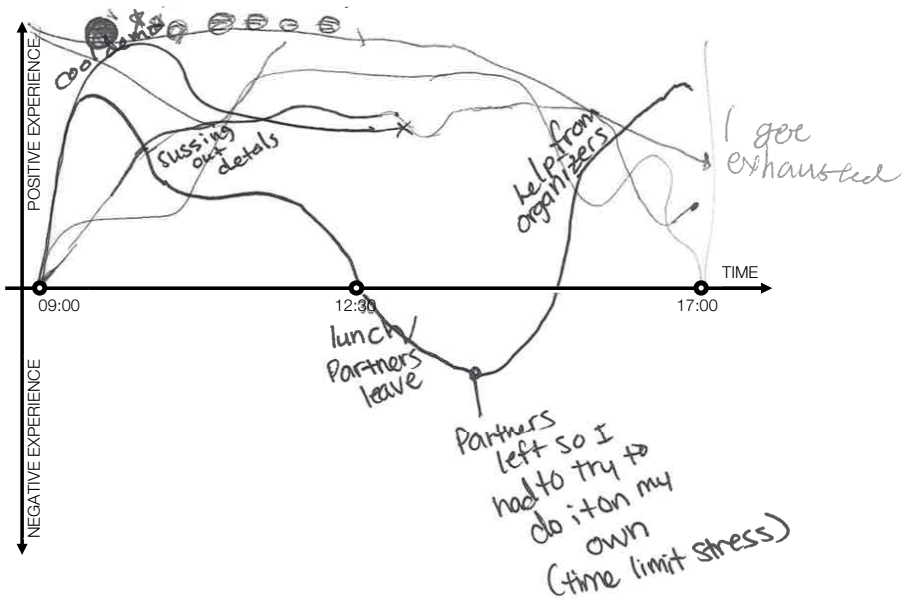


Fig. 3.44 UX curves drawn by the the participants of the workshop.  $n = 5$ .

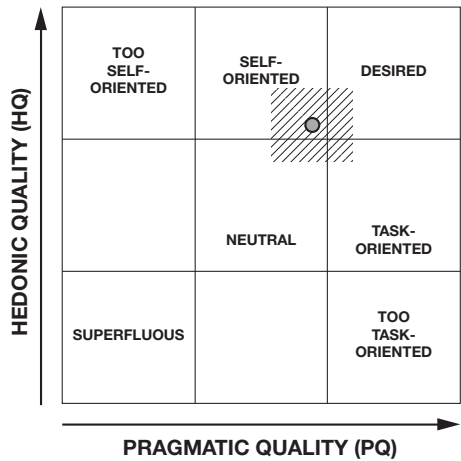


Fig. 3.45 Results of the AttrakDiff questionnaire. The hatched area represents the confidence rectangle.  $n = 5$ ,  $PQ = 0.77 \pm 0.60$ ,  $HQ = 1.21 \pm 0.54$

related to the visual programming software. Minor bugs, software quirks, and usability issues (for instance, a save button had not yet been implemented) resulted in a frustrating experience at times. This is reflected in the lower score for pragmatic quality of the AttrakDiff questionnaire.

The visual programming software was implemented so that the script's logic is executed



in the browser, which then periodically sends movement commands to the robot. This proved troublesome due to issues with connection stability and latency. Since then, the software was rewritten so that scripts are compiled and executed directly on the robot, leading to better responsiveness and stability.

Through the experiment, we identified a number of problems with the architecture of the Blockly API. The API commands, implemented as Blockly blocks, were too verbose, leading to large amounts of repetitive boilerplate code. In future versions, low-level blocks should be grouped into constructs with a higher level of abstraction. This would make for a simpler and more expressive language.

On the mechanical side of the toolkit, we found that high-torque motion, such as moving limbs, proved hard to realize using our system. This is to be expected due to the inherent low torque of the RC servos as well as due to the limited holding force of the snap connectors. This proved to be an issue for the pirate robot, which included an actuated arm, though the problem was mitigated using a liberal amount of hot-melt adhesive. Still, the issues with actuated limbs were expected. The issues could be solved by using better actuators. However, the use of high-end actuators would lead to a dramatic cost increase, and conflicts with the goal of creating an accessible, DIY-friendly platform.

In general, we think the workshop was successful. In the end, all participants succeeded in building a functional animatronic creature in a very short amount of time. Participants praised the format of the workshop in their comments, particularly the open-endedness of the assignment as well as the quick and instinctive prototyping approach. One participant in particular reported that they felt that “failure is ok”, an important prerequisite to the concept of a *sandbox culture*, as popularized by MIT Media Lab. Norman (2002) shows that removing the stress associated with failure can have a tremendous impact on creativity. It causes the brain to switch to a different cognitive state, causing the thought process to broaden.

The results of the workshop show that the toolkit works well as a quick, low-fidelity method for prototyping new social robots. In previous work, our focus was mainly directed toward higher-end, time-intensive prototyping involving CAD modeling and laser-cut embodiment parts. From this, a new challenge arises: designing a prototyping method that fits in-between these two extremes. This “medium-fi” prototyping approach should preserve the spontaneity of cardboard as a prototyping material, yet offer a way to create robots that are more durable and aesthetically more attractive.

### 3.5.2 THE ILLUSION OF LIFE

Expanding upon results of the one-day workshop at TEI (section 3.5.1), we organized a semester-long course assignment based upon the same idea. Twenty 2<sup>nd</sup>-year Industrial Design students were tasked with the design of new social robots using the Opsoro system. This took place over the course of a 12-week semester, as part of one of their Design Studio courses. Students were grouped into pairs, resulting in a total of ten different robots. The

longer time span afforded us the chance to devote more time to the concept, the character design, and the materialization of the embodiment. This was not possible in a one-day workshop.

While these students are not considered design novices, it should also be noted that these students have no expertise in the design of robots. The Industrial Design program is oriented toward general design and engineering. There is no special focus on human-computer interaction or mechatronics design. Considering the students' background, as well as practical constraints, we decided to limit the scope of our experiment to the physical design of social robots. Behavior programming was not part of the course, although this is something we want to include in future experiments.

To aid the students in their design process we imposed a fixed planning that corresponds to the different steps in our platform's methodology. In the first part of the course (week 1-4), all students worked individually. During this time, students worked towards a top-three of robot concepts that could be created with the platform. At this point, we selected the best concept per student, and grouped the students into pairs, with each pair having similar or complementary concepts. From week 5 on, students worked in teams to realize their concept using the platform. To encourage collaboration between teams during the assignment, students were explicitly told that they were allowed to share their own module designs with other teams. This served two purposes: (1) it reduces student workload, (2) it encourages the students to make their custom modules more flexible (i.e. "How can I design my module so that it is not only useful for me, but for other people as well?"). The sections below describe the planning used during the assignment. Figure 3.46 shows one team's process in various steps along this process.

- *Week 1-2: Introduction & inspiration* – Students are given background information on the HRI field, along with examples of existing social robots. During the introduction lesson, students were also given a presentation on the design process of Probo and Ono.
- *Week 3-4: Concept generation* – During this phase, students worked individually toward a concept for a social robot. This process entails the design of an identity and personality that is linked to the functionality and appearance of the robot. After selection of the concepts, the students continued to work in teams of two.
- *Week 5: Quick & dirty mockups* – Starting from sketches, students created rough 1:1 scale foam models for the appearance of the robot. These foam mockups served a number of purposes: they allowed the students to quickly fine-tune the appearance, they provide an indication of the stability and the shape, they can be used to test-fit the modules, and they can be used as a basis for the skeleton design and skinning of the robot.
- *Week 6-7: Modules and skeleton design* – Once the general appearance of the robot is established, a rigid frame needs to be designed to affix the modules, electronics and skin to. To create the design of the skeleton, students were first instructed to create

a digital 3D model of their physical mockup. We recommended the use of sculpting software (e.g. MeshMixer<sup>11</sup>), though some students created their model with NURBS surface modeling tools. For the first design iteration of the frame, the 3D model was converted into slices using Autodesk's 123D Make<sup>12</sup> and then cut out of 3 mm cardboard using a laser cutter. After test fitting the components, the slices were transferred to a dedicated mechanical CAD software package, which was used to further detail the skeleton. This detailing includes adding snap connectors for interconnecting the different parts of the skeleton and for connecting the modules to the skeleton, adding openings for cable routing, and adding reinforcement parts. This second iteration was then again cut from 3 mm cardboard, assembled, and tested by the students. Cardboard was chosen for the first two iterations because it is inexpensive (both in terms of material cost and machine time) and because the cut pieces can be easily modified afterwards using simple hand tools (e.g. scissors, knives, glue, tape). Once the frame designs were finalized, the designs were cut from 3 mm ABS plastic. Nearly all skeletons could be cut from three sheets sized 600x450 mm, which was the number of sheets we provided per team. Any additional sheets had to be provided by the students themselves. This encouraged students to design their frames with efficient material use in mind. During this phase, students also started work on any custom modules they might require for their robot. Most custom modules were simple modifications of existing modules, though some groups also designed modules from scratch. As mentioned earlier, teams were explicitly permitted to share their modules with each other. This resulted in an interesting dynamic where modules were traded between groups (e.g. "We will design module A if you design module B, then we both can use the modules in our robot.").

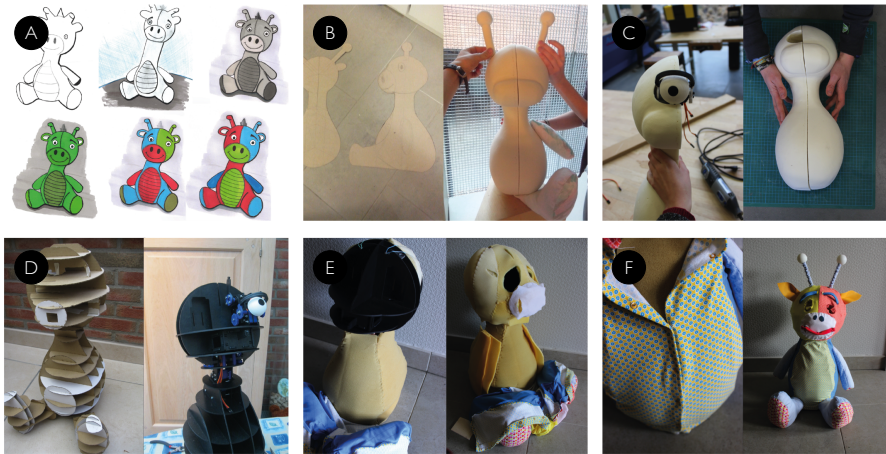
- *Week 8-9: Skinning and facial features* – Once the frame and modules of each robot are finalized, an outer, aesthetic layer needs to be created. Techniques used by the students during this phase varied greatly. One method we suggested was to make a soft padding layer using sheets of soft PU foam, and to then cover this padding layer with an outer, visible layer made from stretchable fabric, such as Lycra. This approach was also used in the design of Ono and works well for soft robots that are created to interact with children. However, seeing that the students' concepts are quite diverse, most teams deviated from this approach quite significantly. While most teams continued to rely on fabrics and textiles, some experimented with radically different techniques, which was especially interesting to us. During this phase, the foam mockups again proved to be very useful, as they allowed the students that used fabrics to easily create patterns for the textile by pinning cut pieces of paper onto the foam mockup. These paper patterns could then be transferred to the textile to cut out.
- *Week 10-12: Skinning, module integration, final adjustments* – During the last weeks of the course, students were mostly working on finishing their designs. Most still required some time to integrate the modules into the skin of their robot.

<sup>11</sup>Autodesk MeshMixer – <http://www.meshmixer.com>

<sup>12</sup>Autodesk 123D Make – <http://www.123dapp.com/make>

- *Week 13: Deadline* – Students presented their work in the first week after the end of the course. Deliverables included (1) a presentation showing the concept, design process and intended interaction scenario, (2) 3D design files, and (3) a set of pictures of the robot depicting each of Ekman's basic emotions (1992).

In addition to the deliverables, students were required to hand in their robots at the end of the course (which was ultimately used to grade the students). We also asked the students to fill in a questionnaire to evaluate the process of the course assignment. This questionnaire comprised questions regarding the use of modules and skinning techniques, (2) questions concerning the difficulty of each phase, and (3) questions regarding subjective appreciation of the platform. After the final presentation, all students completed the questionnaire, resulting in two responses per robot. The questionnaire was not anonymized, so that data of team members could also be compared to each other. The questionnaire was taken in Dutch and the results were translated to English by the authors.



**Fig. 3.46** Design process overview of one robot. (A) Concept sketches, (B) foam mockup, (C) Module test fitting, (D) 1<sup>st</sup> and 3<sup>rd</sup> skeleton iterations, (E) skinning – foam padding, (F) skinning – outer textile layer.



**Fig. 3.47** The ten robots designed during the student course.

### 3.5.2.1 RESULTS

The results of the course assignment are shown in figure 3.47, showing the ten robot embodiments designed by the student teams. Table 3.9 gives a summary of the intended functionality of each robot, the modules they used, and the materials used to make the outer skin.

In general, nearly all module modifications had one goal in common: they aimed to make the module smaller in order to be able to integrate that specific facial feature into their robot. This tends to be a trade-off between size and functionality. The standard modules are usually bigger than their modified counterparts, but offer more DOFs. However, these extra DOFs are not always required to enable the intended interactions.

Many groups modified the length of the levers on the eyebrow and mouth modules to accommodate the outer shape of their robot. This is a very minimal modification and is therefore not included in table 3.9 under “modified modules”. Groups B, G, and H used a modified eyebrow module as a mouth due to space restrictions in their design. These modules only have two DOFs. Consequently, they do not allow the mouth to open and close. Groups A, D, and G all use two eye-eyebrow modules. This module is an amalgamation of the standard eye and eyebrow modules, and is much more compact than the two separate modules together. The eye-eyebrow module has 3 DOFs, as opposed to the 3+2 DOFs of the standard modules. The eyeball is static; only the eyebrow and eyelid are actuated. Group J used two modified eyebrow modules – each with one servo removed – to actuate the bunny ears.

Name <sup>a</sup>	Concept description	Standard modules <sup>b</sup>				Modified modules <sup>b</sup>				Custom modules	DOFs	Skin materials
		E	EB	M	J	E	EB	M	J			
TwinWin (A)	Telepresence system to communicate emotions between two friends, family members, or lovers.	-	-	1	-	2	2	-	-	Neck	10	Stretchable fabric, soft foam, stuffing
Professor Knowall (B)	Teaching (homework) system, to be used in conjunction with tablet for interactive quizzes.	2	2	-	-	-	1	-	-	-	12	Soft foam, EVA foam, hard plastic, stuffing
ReminderBot (C)	Planning and timekeeping aid for people with dementia, autism, or other memory-affecting disorders.	2	2	-	2	-	-	-	-	-	12	Stretchable fabric, soft foam
Kanga (D)	Motivator/coach to stimulate motor function exercises in children with Down syndrome	-	-	-	-	2	2	1	-	Neck	10	Non-stretchable fabric, soft foam, stuffing
Mumble (E)	Encourages tolerant behavior in children. Gradually climbs out of its box as children get to know the robot.	2	2	-	-	-	-	-	-	Lift	11	Non-stretchable fabric, soft foam
DriveMe (F)	Co-pilot for people that spend a lot of time driving. Aids in navigation, communication, and general car functionality.	-	-	-	-	-	-	-	-	Turntable, 2 × LED eye, 2 × Ear.	3	Stretchable fabric, hard plastic
Pillo (G)	Physical affection robot for adults. Inspired by phenomenon of lonely adult men in Japan.	-	-	-	-	2	3	-	-	-	8	Stretchable fabric, non-stretchable fabric, soft foam, EVA foam, hard plastic, stuffing
Poco (H)	Musical coach to encourage children to do their daily musical instrument exercises	2	2	-	2	-	-	1	-	-	13	Stretchable fabric, soft foam, EVA foam
Walu (I)	Replacement for preschool class pets. Supports class activities and teaches children to care for animals without risk to animal wellbeing.	2	2	-	2	-	1	-	-	6 × LED dome	12	Non-stretchable fabric, soft foam, stuffing
AniHero (J)	Clumsy hero with good intentions that tries to encourage children to help with small household tasks.	-	2	1	-	2	2	-	-	-	15	Stretchable fabric

<sup>a</sup> The letters in parentheses refer to the robot pictures in figure 3.4/7  
<sup>b</sup> The letters indicate the module type: eye (E), eyebrow (EB), mouth (M), joint (J).

Table 3.9 Overview of robot concepts, module use, and skinning techniques

In addition to the module modifications, some groups also designed custom modules to be used in their robots. Groups A and D created a neck module to tilt the head of their robots. The module is based on a two DOF neck prototype that was designed as part of a master thesis (discussed in section 4.1.1.5). The pan mechanism was eliminated and the overall design was refined to correspond to the rest of the system. The mechanism uses the same type of servo as the rest of the platform, but relies on a 3D-printed lead screw for mechanical advantage. Group F created a turntable module to be able to turn their robot horizontally. This one DOF module is comparable to the neck module, except that it enables panning instead of tilting. The module is based on a Lazy Suzan bearing, with the servo transmitting motion via an internal gear. The course assignment yielded a number of cases where existing modules fell short, and thus had to be modified or replaced. The work done by students on the modules is a valuable source of inspiration for future elements of the platform and proves the easy adaptability of the platform.

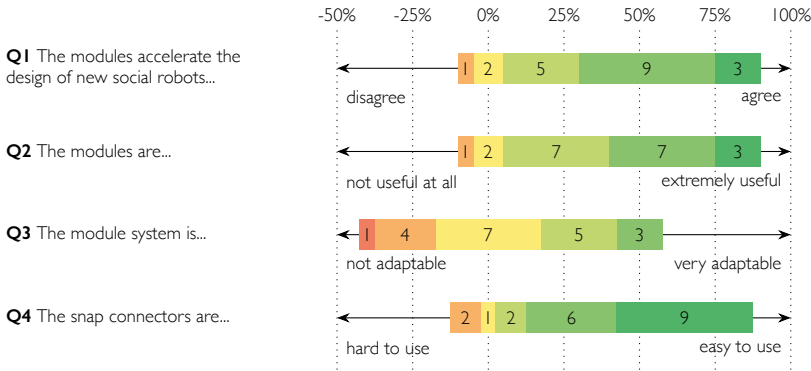
As mentioned earlier, students experimented with completely new techniques and materials to create a skin for their robot embodiment. During the orientation presentations, we proposed a skinning method to the students, which involves covering the skeleton with a soft foam-padding layer, which is then covered with an outer layer made of stretchable fabric. However, big differences in the intended modes of interaction of the students' robots lead to different priorities for the skin design. To elaborate: Ono was originally intended as a huggable robot for children, much like Probo. A soft, huggable embodiment was therefore essential. On the opposite side of the spectrum are robots such as DriveMe (fig. 3.47 F). They are not intended to be touched, so a hard plastic exterior is a valid solution. A third example is Professor Knowall (fig. 3.47 B), which falls somewhere in between the two. The robot is intended for interaction with young children, thus a cold, hard exterior would not be appropriate. On the other hand, the robot does borrow the connotation of a professor to create a sense of distance between the child and the robot, so a soft foam exterior would also not have been an appropriate choice.

Within the class group, team H pioneered the use of Ethylene-Vinyl Acetate (EVA) foam. This thermoplastic foam can be formed into three-dimensional shapes through thermoforming, a technique where thermoplastic sheet material is heated and pulled over a mold with the aid of a vacuum. This results in semi-flexible, thin parts that are soft to the touch. A number of groups also thermoformed PS sheets, most of these groups combined the rigid PS parts with an outer layer of EVA foam. One notable exception is group G, which relied solely on thermoformed PS for the majority of the exterior. Group I experimented with the use of felt. This material can be formed into three-dimensional shapes with the aid of steam, but has a tendency to fray, resulting in a messy look.

The goal of our questionnaire was to gain insight as to how the platform and the design process were perceived by the students. Whereas data on the modules and skinning techniques are represented in the robots themselves, it does not allow us to gauge the potential difficulties in the process. The first part of this questionnaire attempts to measure the general sentiment of the students toward the platform. This part comprises four 7-point Likert scale statement questions ( $M_4$ ):

- "The modules accelerate the design of new social robots..." disagree / agree.
- "The modules are..." not useful at all / extremely useful.
- The module system is..." not adaptable / very adaptable.
- "The snap connectors are..." hard to use / easy to use.

Results of these questions are show in figure 3.48. The nature of the experiment and of the platform itself makes it difficult to compare data to any baseline. However, in our opinion the data does show a generally positive trend, with the averages of each question being in the desirable end of each scale. For the second part of the questionnaire, we asked students to rate the level of difficulty of each phase of the design process. 7-point Likert scales were used ( $M_4$ ), with a rating of 1 indicating that the phase was easy, and a 7 indicating that the phase was hard. Results are shown in figure 3.49. As expected, the students perceive later phases as more difficult than the phases in the beginning of the design process. There is a twofold explanation for this phenomenon. Firstly, as part of their other courses, students are intimately familiar with general design phases such as concept generation, foam modeling, and cardboard prototyping. On the other hand, later phases of the design process simply pertain more engineering work and involve more specific knowledge.

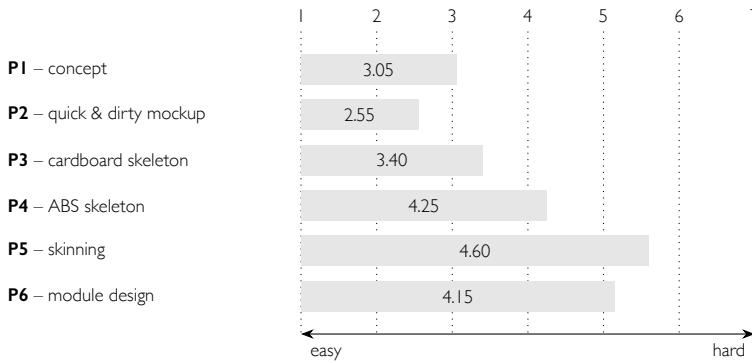


**Fig. 3.48** Results from the Likert scale questions.  $n = 20$ .

### 3.5.2.2 DISCUSSION

One of the most noticeable shortcomings of our platform in this experiment is the adaptability of the modules themselves. Whereas the results indicate that the platform is flexible on a toolkit-level, this flexibility does not translate to the module-level. This observation is also supported by the result of Q3 of the questionnaire (fig. 3.48). We note a trend in the modifications that the students made to the modules: typically, these modifications were made to make the modules more compact at the expense of reduced functionality (e.g. eye-eyebrow module, eyebrow module as mouth). Consequently, future versions of the





**Fig. 3.49** Relative difficulty of each phase.  $n = 20$ .

platform should include multiple alternatives for the same facial feature. Students also frequently changed the length of the levers of the mouth and eyebrow modules. It is evident why this modification is so common: the geometry of the robot's face directly influences how far the levers need to reach. Within this study, the lever design files were modified manually. However, future versions of the platform should anticipate this change. In general, future module versions should be less prescriptive in their intended use and should be easier to modify and hack.

Two groups (fig. 3.47 F & I) incorporated LEDs in the design of their robot. In previous experiments, we worked with children with ASD. In this context, displays and LEDs are a hindrance to social interaction, hence why they were avoided in the design of the platform. However, the two student designs did show the merits of using LEDs to increase the expressive range of robots in their respective contexts. Consequently, support for addressable RGB LEDs (i.e. NeoPixels) has been added to the latest iteration of the platform's electronics.

The actuation of very large or very strong mechanisms (e.g. arm, neck) proved to be a third point of friction during the experiment. As discussed in section II, the Opsoro platform is built around low-cost hobby RC servos. While these servos are more than sufficient for the actuation of facial features, they are much less suitable for large or strong movements. Students used multiple strategies to work around this problem. Some groups (fig. 3.47 A, D, E) employed a 3D printed screw mechanism to gain enough mechanical advantage to move their neck. Others (fig. 3.47 C, H, I) attached lightweight, flexible arms to the end of joint modules. This way, if a user pushes down on an arm, the arm itself bends, and the servo is protected from excessive torque. A final case of note is Pillo (fig. 3.47 G). Originally, the robot was intended to have arms so it could embrace the user. In the end, the arms proved to be too troublesome, so instead the team opted to eliminate arms and instead focus on adapting the shape of the torso in order to insinuate and stimulate hugging behavior.

The experiment also revealed a number of noteworthy aspects concerning the methodology

used in class. First of all, using a limited number of shared modules yielded interesting effects, both positive and negative. On the plus side, by limiting the available modules to two “full” toolkits, we forced the different student groups to collaborate. The result of this approach was that the students’ module designs tend to be generic and much less bound to a single specific robot. Most of these new modules were used for multiple robots. A second advantage is that students had to take the (dis)assembly into account, seeing that they would have to add/remove the modules many times over the duration of the course. The main downside of this approach is that the limited number of modules ended up being a bottleneck in the design process, seeing that much time was lost by disassembling and reassembling robots. A better balance needs to be found between the number of groups and the number of available sets of modules; two complete sets for ten groups is simply too little.

We also noticed that our approach to alternate between low-fi (foam and cardboard mock-ups) and high-fi (laser cutting) prototyping worked out well. Our impression is that this encourages students to fail early and often. Potentially fatal problems are thus caught much sooner in the design process. The way students used their foam mockups to test the size and position of modules is an example of this. While the benefits of iterative design are well known within HCI, the complexity of robotics design makes it tempting to use a waterfall design approach, where a robot is designed, built, and tested in a single iteration, meaning that mistakes are only discovered at the end. We have observed that our method avoids this pitfall.

Thirdly, the questionnaire results indicate that the ABS skeleton phase, the skinning phase, and the module design phase are experienced as the most difficult parts of the design process (fig. 3.49). As mentioned earlier, the students’ designs pointed out a number of shortcomings in the current selection of modules in the platform. We hope that an expanded set of modules will eliminate the need for custom designed modules in most cases. With respect to phases 4 and 5, we believe that a software tool, in the form of a specialized CAD program or plugin, could simplify these phases significantly. The design of a skeletal frame involves a large amount of work to draw up in CAD, but much of this work is completely formulaic in nature, requiring very little creativity, and would be an ideal candidate for automation. We envision this software as a more advanced version of 123D Make, where the user could load a 3D model of the outer shape, and then input the position of each required module. The software could then generate a skeleton for that specific embodiment, automatically adding features such as snap connectors, part numbering, and module mounting locations. Finally, dividing and flattening the outer shell, similar to UV mapping used in computer graphics, could easily generate skin and foam patterns.

### 3.5.3 CLASSROOM OF THE FUTURE

The experiment described in this section took place during the “Classroom of the Future” event at the 2016 Frankfurt Book Faire. The purpose of the experiment was to evaluate a simplified version of the Opsoro system with secondary school students. Over the

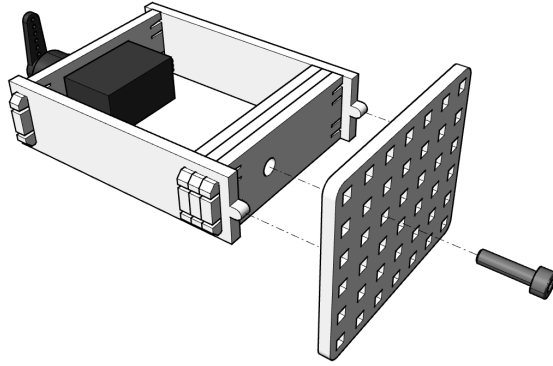
course of a two-hour workshop, students created and programmed custom robots using cardboard, craft materials, and Opsoro modules, as depicted in figure 3.50.



**Fig. 3.50** Participants building custom robots using the Opsoro modules and craft materials.

In the experiment, a different connection system for the modules was used. This system – dubbed the *Opsoro Grid* (fig. 3.51) – does not rely upon snap connectors, unlike previous versions. Instead, modules can be connected to a grid plate using a single screw. The holes of the grid plates are spaced 8 mm apart, making the system compatible with LEGO Technic bricks. The modules contain two locating pins in addition to a single threaded hole. The locating pins are used to position the module on the grid, preventing rotation. A screw is then used to fasten the module to the grid, locking the connection. The single-screw connection bears resemblance to the hybrid connector system of the Robot Blocks experiment (section 3.2.2.3), although this connection system is designed for a different type of application and uses a different fabrication process.

During the experiment, participants worked together in small groups (2-5 person teams) to design and program their robots. To this end, we provided four kits, so that four groups could work in parallel. The experiment ran over the course of a day, with each team taking approximately 2 hours to complete a robot. The kit consists of a set of electronics, five modules, and a large grid plate. Keeping in line with the efforts to simplify and reduce costs of the Opsoro system, we chose to use smaller, simplified modules, as informed by the results of previous experiments. The modules used during this experiment were smaller, had less DOFs, and were built using micro-sized servos (as opposed to standard-sized



**Fig. 3.51** Grid connection system used in the Frankfurt experiment

servos).

To design robots, participants first positioned and affixed the modules on the grid plate. Then, they created cardboard “skins” to go over the modules and backplate. The grid system afforded participants the ability to position modules wherever they wanted. The less prescriptive design of the modules in combination with the low weight of the materials enabled functionality such as moving arms and ears. Figure 3.52 shows some of the embodiments that were created during the experiment.

The robots were programmed using the Blockly-based Visual Programming app. The Opsoro Blockly API was updated to address some of the problems that we encountered in previous experiments. Additionally, an early version of the online community platform was tested. The online platform is similar to the robot’s web interface, and allows scripts to be created and tested in the browser using a virtual robot simulator.

As part of the evaluation, participants were asked to fill in a survey. The large number of participants, the limited amount of time, as well as the language barrier meant that a paper survey was the best evaluation tool for the experiment. The questionnaire, translated in German, consisted of the following questions:

- General personalia questions: name, age, gender.
- 7-point Likert scale statements ( $M_4$ ), with a value of one indicating complete disagreement, and seven indicating complete agreement:
  - “I could build what I wanted to build.”
  - “The connection system is easy to use.”
  - “The modules are adaptable.”
  - “I like the aesthetics of the Opsoro system.”
  - “I like the functionality of the Opsoro system.”

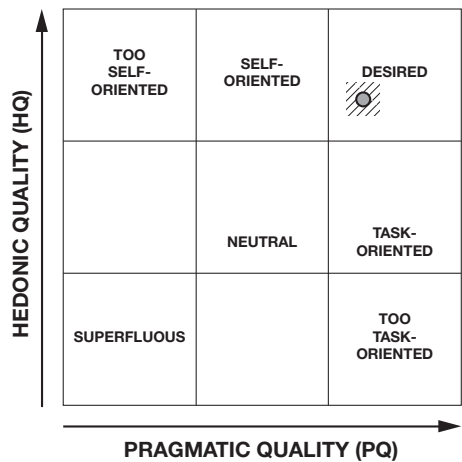


**Fig. 3.52** Cardboard embodiments designed by the participants.

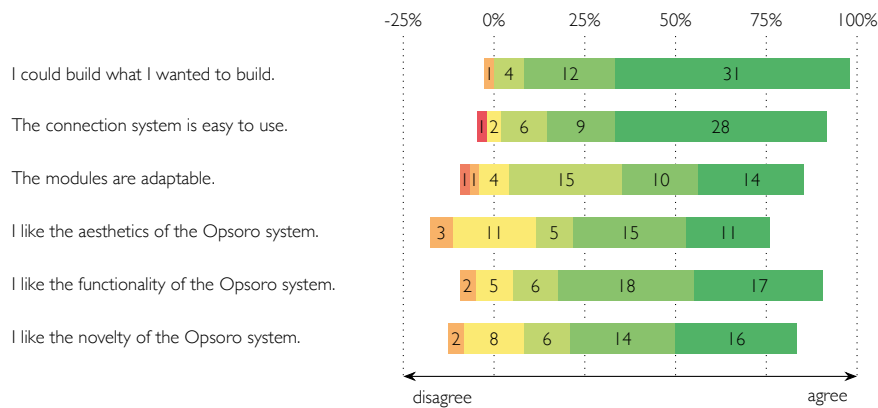
- “I like the novelty of the Opsoro system.”
- The AttrakDiff-Short questionnaire ( $M_2$ ). This variant of the AttrakDiff questionnaire consists of only ten antonym-pairs, as opposed to 28 antonym-pairs in the full AttrakDiff questionnaire.
- Open questions ( $M_6$ ):
  - “Which workshop aspects did you like?”
  - “What was the most annoying aspect, or where did you experience the most problems?”
  - “What did you learn from the workshop?”

In total, 48 workshop participants filled in the survey. The average age of respondents was 17.35, with a spread of  $\sigma = 1.86$ . The youngest respondent was 14, the oldest 23. Respondents were predominantly female (66%), with 32 girls versus 16 boys. Figure 3.53 shows the result of the AttrakDiff-Short survey. Two incomplete surveys were discarded, resulting in a sample size of  $n = 46$ . The latest iteration of the Opsoro system is situated firmly in the category of desirable products, a notable improvement over previous experiments. With a value of 1.84, the overall attractiveness of the system ( $ATT$ ) was also rated fairly highly. The large number of participants resulted in a narrower, more precise confidence interval, as indicated by the hatched area. Figure 3.54 shows the results of the six Likert scale questions.

The results of the AttrakDiff-Short questionnaire position the Opsoro Grid system firmly within the quadrant of desirable products, a direct improvement over previous experiments where the AttrakDiff tool was used (e.g. section 3.2, 3.4.3 and 3.5.1). The results of this experiment also show a tighter confidence rectangle than previous experiments, owing to



**Fig. 3.53** Results of the AttrakDiff-SHORT questionnaire. The hatched area represents the confidence rectangle.  $n = 46$ ,  $PQ = 1.52 \pm 0.26$ ,  $HQ = 1.64 \pm 0.24$



**Fig. 3.54** Results from the Likert scale questions.  $n = 48$ .

the larger number of participants. Still, one should be careful with direct comparisons to previous test results. Previous tests had different workshop objectives, different user groups, and different durations. Another point of nuance is that respondents only had a limited timespan to interact with the system, and novelty and limited scope of the experiment may have been factors in the favorable results of the test.

In the feedback from the open questions, we noticed that “creativity” is frequently mentioned as a positive aspect of the Opsoro system. Of the 41 respondents that filled in the open questions, 20 mentioned “creativity” in their responses (48.8%), 10 mentioned fun (24.4%), and 7 mentioned technology (17.1%). Multiple respondents also said that building robots is not as hard as it appears, and that it is fun to combine technology with creativity. As expected, negative feedback was mostly related to technical problems, such

as defective servos and problems with the audio quality and the volume of the speaker. Comments from the open questions also hint at the potential of Opsoro for STEM education, as indicated by remarks such as “*Technology does not have to be boring*” and “*You can do many cool things with technology*”. With the predominantly female sample, we think the reported positive experiences also hint at opportunities to inspire girls to engage with technology and STEM education.

Overall, results from the Likert scale questions (3.54) were also generally positive. The adaptability of the modules is rated remarkably higher than in the previous experiment (fig. 3.48). Results also hint that the grid system is preferred over the snap connector system. From personal experience, we found that the grid system is much more rigid and secure than the snap connectors. One point of improvement is the aesthetics of the system, which is ranked the lowest in the Likert scale questions. Cardboard is a relatively low-fidelity prototyping material. Even though the material can be manipulated easily and quickly, it is difficult to create detailed, high-quality embodiments. Alternatives should be explored in the future, perhaps with pre-made laser-cut shapes or with thermoformed plastic shells.

### 3.5.4 SUMMARY

The Opsoro toolkit integrates insights from previous iterations into a real platform that provides DIY tools and a design method ( $G_1$ ) enabling users to create custom social robots. In addition to the physical and virtual tools, the design flow for the users is also evaluated. The results show that easy-to-use building blocks can foster creativity by addressing the right skills and by matching them to the right challenges. We have investigated low-fi prototyping techniques that deliver fast results (section 3.5.1), as well as high-fi prototyping techniques that lead to better, more durable creations (section 3.5.2). The last iteration (section 3.5.3) shows great potential with respect to “medium-fi” prototyping, maximizing creativity with technological building blocks that can bring custom social characters to life.

With these iterations we gained new insights into finding the perfect balance between the ease of use ( $G_2$ ) of the system and the potential for creating emotionally expressive ( $G_3$ ) characters ( $G_5$ ) with facial features ( $G_4$ ). The building blocks are not prescriptive and offer the potential for custom creative interpretations ( $G_7$ ) while maintaining a positive flow experience ( $G_6$ ) by matching different skills and challenges. Within the last iteration, an early version of the online platform ( $G_{10}$ ) was also tested. Preliminary tests with the online platform show promise, allowing scripts to be tested virtually before running them on robot hardware. Furthermore, the last iteration in particular (section 3.5.3) has shown that the creation of social characters was described as a positive experience by girls, enabling better gender balance in future STEAM workshops ( $G_8$ ).

### 3.6 CONCLUSION

In this chapter, we discussed the iterative design process that ultimately led to the Opsoro platform. We have substantiated our design decisions with results from experiments as well as references to literature. This chapter has described five distinct yet interrelated design tracks, illustrating the non-linear nature of a human-centered design process. Table 3.10 gives an overview the iterative design process. We note that a very large variety of designs can be created with a small set of maker-centric tools and techniques. Each of the designs described in this chapter could be built at any reasonably equipped FabLab.

We also note that in each of our experiments, the participants succeeded in building robots from scratch, despite a lack of knowledge and experience. A recurring theme is that they achieved these goals despite initial reservations. Finally, we also found that the participants quickly developed a bond with their creation, similar to the IKEA effect (Norton et al., 2012). This phenomenon occurred when participants built a “standard” Ono, but its presence was even stronger when they designed their own embodiments from scratch. This chapter has focussed on discussing the iterative design process, mentioning technical aspects of the system in passing. The next chapter continues with an in-depth description of the technical implementation of the Opsoro platform.

Design goal		Hexapods	Robot Blocks	Ono gen. 1	Ono gen. 2	Opsoro
$G_1$	Open	✓	✓	✓	✓	✓
$G_2$	Easy to build	✓	✓	✓	✓	✓
$G_3$	Emotional expressiveness			✓	✓	✓
$G_4$	Facial features			✓	✓	✓
$G_5$	Character			✓	✓	✓
$G_7$	Creativity		✓	✓	✓	✓
$G_6$	Flow		✓			✓
$G_8$	Diverse knowledge domains		✓			✓
$G_9$	Low cost	✓	✓	✓		✓
$G_{10}$	Community-friendly	✓	✓		✓	✓

**Table 3.10** Overview of the iterative design process



## Chapter 4

# OPSORO: OPEN PLATFORM FOR SOCIAL ROBOTICS

The previous chapter has shown how the Opsoro system makes it possible to build a wide variety of social robot embodiments by combining standardized modules with a custom skeleton and appearance. Digital manufacturing technologies are used to facilitate this process by providing a fast, accessible method for producing a high-fidelity skeleton. In addition, because the modules are built using the same techniques, users are free to make their own modifications to the modules as they see fit.

Typically, modular robots are comprised of stackable, tilable or otherwise repeatable units. Examples include the Distributed Flight Array (Oung and D’Andrea, 2011) and roBlocks/Cubelets (Schweikardt, 2011). In the Opsoro system, modularity is approached in a different way. The complex subsystems of a social robot are encapsulated in reusable modules. These modules are not attached directly to one-another. Instead, they rely upon a custom frame and skin in order to form a functional whole. This way, the flexibility of custom designs is combined with the time-saving benefits of standardized modules. One can draw a comparison between our approach toward modularity, and the reality of the natural world. Animals (and humans) can have wildly varying appearances, though certain facial structures look very similar across many species. For example, the size, position, and orientation of eyes varies across species, though the basic spherical shape, the iris, and the eyelid remains similar.

Our approach toward modularity also incorporates aspects of an “untoolkits” (detailed in section 1.1.5). D. A. Mellis, Jacoby, et al. (2013) posit that the design of traditional toolkits often has a tremendous impact on the form and aesthetics of artifacts. Instead of packaging all functionalities in high-level building blocks, untoolkits provide tools and techniques leverage existing components in new ways. While our approach uses high-level modules, they are combined together through a design method that relies upon digital

manufacturing techniques. In this sense, the Opsoro toolkit is more than a set of blocks, it also incorporates techniques to leverage these blocks in novel ways.

This chapter gives an in-depth overview of the platform's technical implementation: including the mechanical hardware design, the electronics implementation, and the software architecture. The chapter is structured as follows:

- **SECTION 4.1** details the mechanical aspects of the platform, including the design of the modules and a description of the embodiment design methodology.
- **SECTION 4.2** discusses the architecture of the Opsoro electronics. The section elaborates upon the different iterations of the robot controller board, discussing the reasoning behind the technical decisions. This section also has background information on the use of sensing human touch.
- **SECTION 4.3** talks about the software implementation of the Opsoro platform. This includes the model and algorithms used for synthesizing facial expressions, web application design, and the app ecosystem.

## 4.1 HARDWARE

The facial expressions of the human face are the result of a complex interaction between various muscles, bone structures, and different layers of soft tissue. The Facial Action Coding System (FACS) (Ekman et al., 1978) defines over 40 different Action Units (AUs) in the face. In the FACS system, each AU represents a fundamental action of a muscle or group of muscles. Replicating all the possible motions of the human face in a robotic agent is a daunting task made difficult by the sheer number of actuators in a small area, as well as the challenge of simulating human tissue with synthetic materials. The work of Ishiguro (Nishio et al., 2007; Ogawa et al., 2012) represents one of the most comprehensive efforts toward the goal of replicating human facial expressions. Still, even in these robots, there are fewer facial “muscles” compared to the human counterpart. In this work, a different approach is used. Instead of aiming to replicate human-like facial expressions, we opt for an abstract, iconic approach, inspired by the works of fiction from companies such as Pixar and Disney (Johnston and Thomas, 1995).

In essence, the system needs to strike a good balance between the level of social expressiveness ( $G_3$ ) and the cost ( $G_9$ ) and complexity ( $G_2$ ) of the mechanisms enabling that expressiveness. We opted to forego exact imitation of human expressiveness for two reasons. First of all, such complexity has a prohibitively expensive cost associated with it, counteracting the goal of making social robotics research and applications more accessible ( $G_1$ ). Secondly, such an endeavor has a high risk of ending up in the Uncanny Valley (Mori, 1970), ultimately impeding the interaction between humans and robotic agents. Instead, we focus on iconic, rather than realistic representation of emotion.

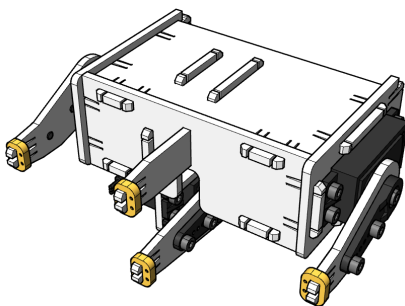
## 4.1.1 MODULES

In the system, the various components are grouped into subunits of related actuators and mechanisms, called modules. Doing so has a number of important advantages. To begin, this approach simplifies assembly; modules can be put together outside of the body of the robot, where there is more room to manipulate components and perform assembly steps. Secondly, in case of damage, modules can be removed and replaced or repaired quickly. Thirdly, the modular architecture makes the system ideally suited for accelerating the design of new robots, as complex functionality is packaged in easy-to-use building blocks. And finally, the modules allow for upgrades and customization, enhancing the lifespan and potential usefulness of robots built using the system.

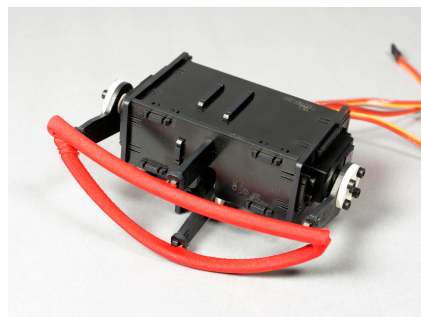
The modules of the Opsoro system use snap connectors as a common interface. Each module encapsulates the functionality of one specific facial/body feature, such as an eye or a mouth. As of yet, the modules focus on actuation only, though it is also possible to incorporate sensors. The next sections give an overview of the modules that are currently available within the system.

### 4.1.1.1 MOUTH MODULE

The mouth module implements mouth motion with three degrees of freedom. The structure of the module is formed out of a laser-cut ABS box frame. The box frame has two 30 mm snap connectors to interface with the frame of a robot, as is standard for all Opsoro modules. Motion is provided by three standard-size RC servos. Two servos are used for the left and right mouth corners. The third servo actuates the middle of the bottom lip. The middle of the upper lip is fixed, though both upper and lower lip can be shaped through the relative motion of the three servos.

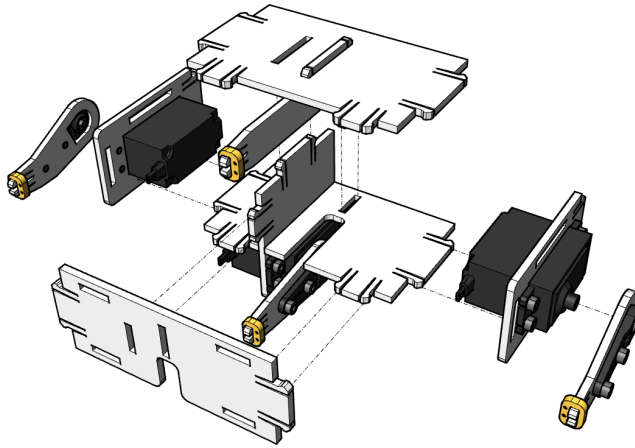


**Fig. 4.1** CAD model of the mouth module



**Fig. 4.2** Mouth module prototype

The levers attached to the servos are connected to the skin of the robot through removable snap connectors, shown in yellow in figure 4.1 and figure 4.3. Four snap connectors are sewn on the inside of the skin at the appropriate points: the two mouth corners, the middle of the upper lip and the middle of the lower lip. The length and the range of



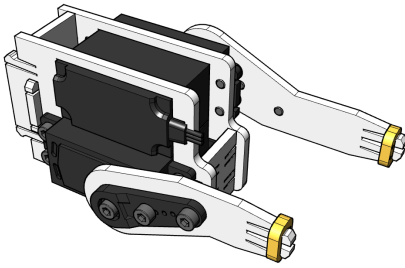
**Fig. 4.3** Exploded view of the module

motion of the levers is dependent on the embodiment design of the robot. The length may be customized by generating a new lever design using the parametric CAD model. The range of motion can be customized in software by modifying the DOF configuration, as explained in section 4.3.1.1.

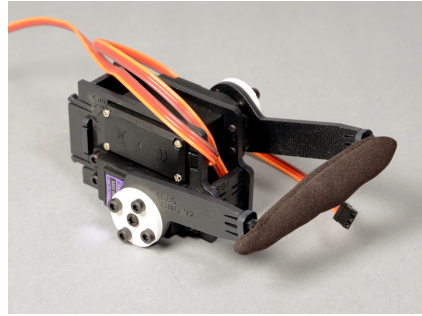
#### 4.1.1.2 EYEBROW MODULE

The eyebrow module provides a functional eyebrow with two degrees of freedom. Motion is achieved through two standard size RC servos connected to the inner and the outer endpoint of the eyebrow. The servos are attached to a laser-cut ABS frame with two 30 mm snap connectors that mate with the robot's frame. As with the Mouth module, the module is connected to the actual eyebrow through a removable snap connector which is sewn onto the eyebrow.

The two standard-sized are stacked on top of each other, with the output shafts pointing at opposite directions. Servo levers with a bent shape are used so that the levers are horizontal and at the same position when the servos are in neutral position. This solution was chosen in order to create a compact, powerful module design with a correct distance between the output levers. As with the Mouth module, output lever length and range of motion can be customized to suit the needs of a specific robot embodiment design. A typical robot would use two eyebrow modules. The module has two variants, a standard (right) version and a mirrored (left) version. This is done to accommodate facial symmetry. Both variants use the same components, but assembly is mirrored.



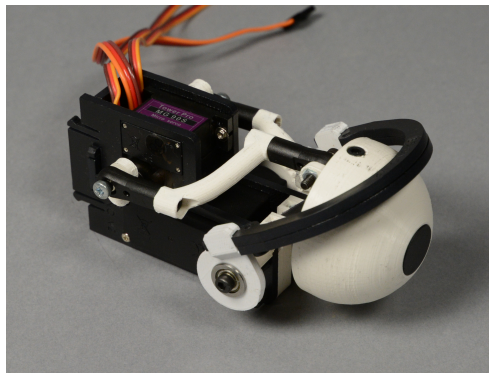
**Fig. 4.4** CAD model of the eyebrow module



**Fig. 4.5** Eyebrow module prototype

#### 4.1.1.3 EYE MODULE

The eye module, shown in figure 4.6, implements a two DOF eyeball with a one DOF eyelid in a single module. Achieving realistic eye movement requires many different moving mechanisms in a compact and limited space. Difficulty is compounded by the fact that the eyeball itself needs to rotate in two different directions around the same point. These reasons make the Eye module the most complex module of the Opsoro platform. As with other modules, many of the structural parts are laser-cut from ABS sheet material, though because of the intricacy of the required mechanisms, the module also incorporates a large number of FDM 3D printed parts.

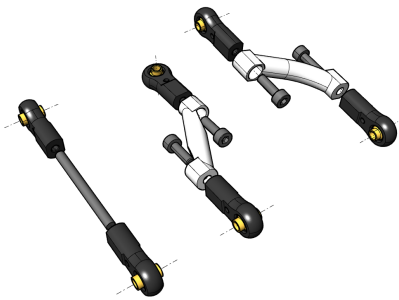


**Fig. 4.6** Eye module prototype

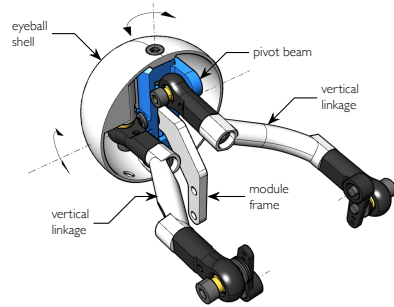
The first major challenge was the implementation of the eyeball that rotates both horizontally (left/right) and vertically (up/down) over a sufficiently large range. These two motions need to be controllable independently from one another. Early versions of the module achieved this through a kinematic chain of two micro-size servos connected in series. Because of space constraints, the centers of rotation of the two servos were not exactly the same. Though minimal, this resulted in a non-negligible amount of translation of the eyeball when it was made to rotate. Additionally, the component linking both servos to-

gether proved to be a weak-point, and the resulting system had a considerable amount of slop.

The second iteration of the eyeball movement relied upon a different mechanism, as shown in figure 4.8. Here, two linkages are used to rotate the eyeball indirectly. Though more complex, the mechanism results in a higher resolution and the centers of rotation coincide. The core component of the mechanism – the pivot beam – has two rotational axes around which the eyeball hinges. The first axis, which enables the eyeball to rotate up and down, is controlled by a linkage connecting the pivot beam to a micro-size servo toward the back of the module. The second axis of rotation allows the eyeball shell to rotate horizontally with respect to the pivot beam, and is controlled by a linkage connecting the eye shell to a second micro servo.



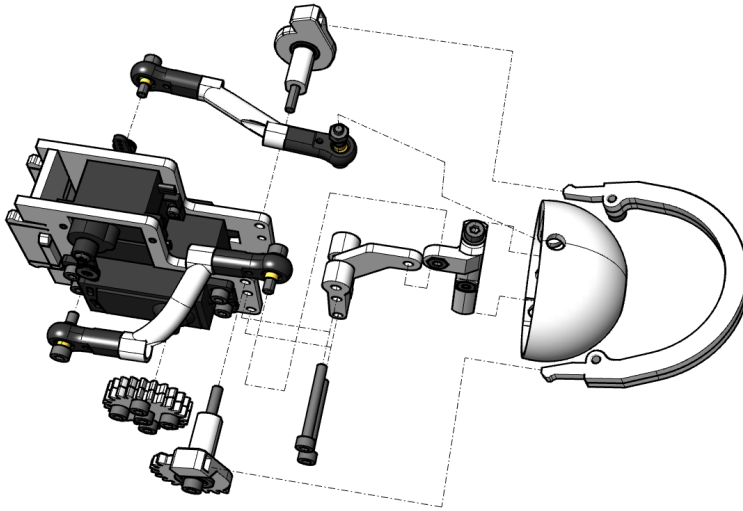
**Fig. 4.7** Ball joint linkages



**Fig. 4.8** Eyeball pivot mechanism

The linkages consist of two RC-grade ball joints that are connected together via a 3D-printed beam. The ball joints are needed in order to avoid binding caused by joint misalignment as a side effect of eyeball movement. Figure 4.7 shows examples of these linkages. The left linkage shows how the ball joints are intended to be used in RC craft, using a piece of threaded rod to connect the two together. The middle and right linkages show how they are used in the eye module. By using a 3D-printed connector piece, curves can be incorporated into the design of the linkage, which is important in the design of the module for avoiding collisions and keeping the size of the module compact. In addition, this approach has the benefit of constraining the distance between ball joint centers, improving assembly repeatability.

In addition to the two DOFs of the eyeball, the module also has an actuated eyelid. This third degree of freedom is powered by a standard size RC servo. The mechanism of the eyelid requires more torque than the eyeball mechanism because the eyelid needs to be able to stretch and deform the foam and textile of the robot's skin. For simplicity's sake, only the upper eyelid is implemented, even though in human anatomy both the upper and lower eyelid move. The module's eyelid is implemented as a rigid arc-like component, as shown on the right-hand side of figure 4.9. This beam is connected to an eyelid flap in the textile skin of the robot, resulting in a completed eyelid that covers the eyeball when closed. The eyelid arch is made to be removable using snap connectors. This way, the arch may first be attached to the skin and then connected to the eye module once the skin is



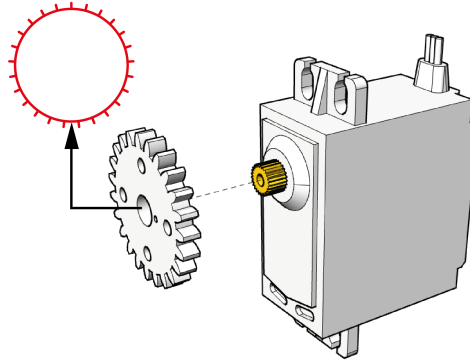
**Fig. 4.9** Exploded view of the module

in place.

The axis of rotation of the eyelid does not coincide with the vertical axis of rotation of the eyeball. The eyelid's axis of rotation is shifted more toward the back of the module so as not to interfere with the horizontal eyeball rotation. The eyelid arch is supported on both sides of the module by a flanged miniature ball bearing. One side of the arch is actuated by the servo, the opposite side is an idler.

A one-to-one gear transmission is used for the eyelid, as shown in the bottom left of figure 4.9. Again, the gear train is needed to avoid collisions and to keep the module compact. The drive gear of the eyelid mechanism is made using the laser cutter. The main challenge here is interfacing the laser-cut gear with the servo's toothed spline. Due to the spline's small and detailed tooth profile, one cannot simply cut out the shape of the spline. Instead, a radial pattern of lines was used to create a surface that meshes well with the profile of the spline, as shown in figure 4.10. The laser beam has the effect of melting the ABS plastic and as a result, the lines of the 2D drawing turn into triangular incisions in the inner surface of the hole. The gear can then be press-fit onto the servo spline, and the elasticity of the ABS plastic compensates for the negative clearance.

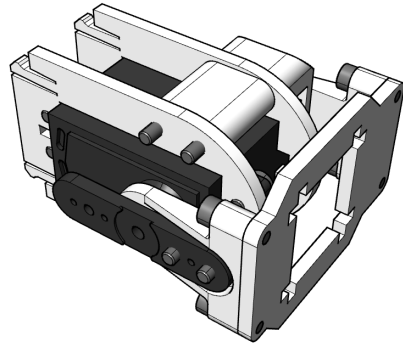
The driven gear of the mechanism is 3D-printed. The part is a gear segment, as only 60° of motion is needed for the functionality of the eyelid. The driven gear is supported by a miniature flange ball bearing, which is press-fit into the part. The part has a rectangular slot which mates with the snap connector of the eyelid, allowing the eyelid arc to be attached or detached easily.



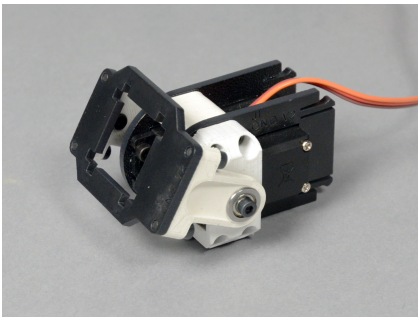
**Fig. 4.10** Connection between servo spline and laser-cut gear.

**4.1.1.4 JOINT MODULE**

The joint module packages a standard-size servo into a one-DOF module to facilitate the use of RC servos with the rest of the system. This module is currently not used in the Ono robot, but was created to support creative usage scenarios in workshops and courses. The ABS-plastic frame of the module has two 30 mm snap connectors, conforming with the rest of the system. The end effector of the module contains the corresponding module mounting pattern, allowing other modules or attributes to be attached. The mounting pattern is designed so that objects can be attached either normally or at a 90° angle, improving the versatility of the module. The module has a male connector on one end and a female connector on the other end, consequently multiple joints can be chained together to form a larger structure.



**Fig. 4.11** CAD model of the joint module

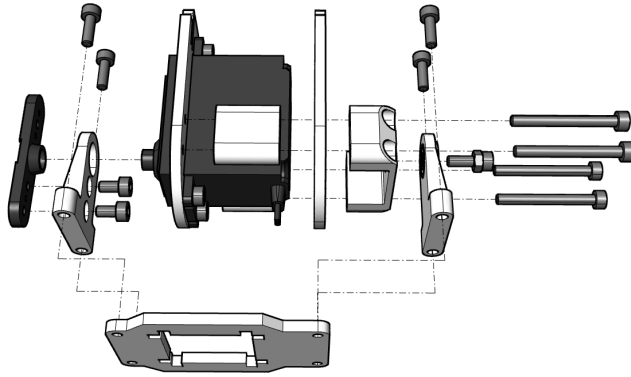


**Fig. 4.12** Joint module prototype

The end effector of the module is supported at both ends. A common problem with RC servos in small-scale robotics applications is the torque load perpendicular to the servo's axis of rotation, “peeling” the horn away from the splined shaft of the servo. This problem was particularly noticeable in the Stigmergic Ant (section 3.1), where the large forces re-



quired to move the robot resulted in misalignment, vibrations, and excessive servo torque loads. The joints of the ant were supported solely by the servo horn, there was no idler on the opposite end to improve stiffness of the joints. Consequently, a significant portion of torque was lost because of lack of rigidity.



**Fig. 4.13** Exploded view of the module

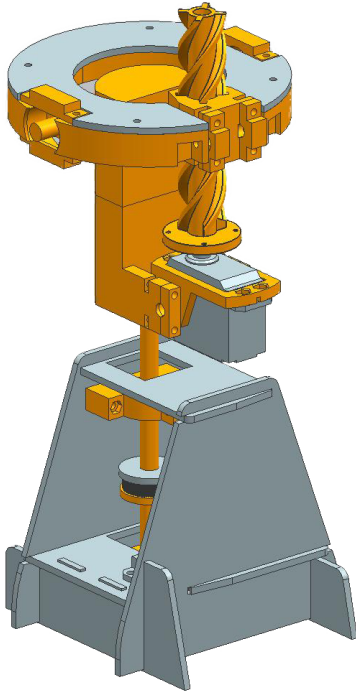
This problem was avoided by incorporating an idler in the design of the joint module. The idler is shown on the right-hand side of figure 4.13. The implementation of the idler uses a 3D printed ball bearing hinge, which is supported by a bracket that is mounted over the end cap of the servo, resulting in a rotational joint that is in line with the axis of rotation of the servo. Some off-the-shelf servos have a threaded hole in the end cap intended for this purpose, though these are hard to find as this feature is often not advertised. For this reason, we opted to use an extra part to support the idler.

#### 4.1.1.5 NECK MODULE

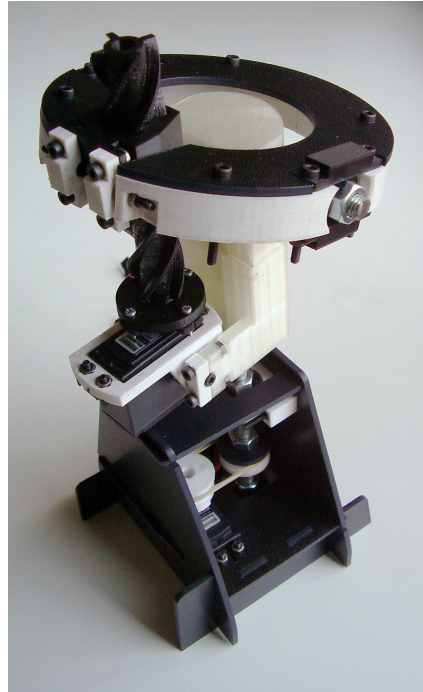
The final module of the system is an experimental neck module. As implied by the name, the purpose of this module is to actuate the head of a social robot. Two different iterations of the module were created in the context of students assignments, based upon our input. The design of a low-cost neck mechanism is challenging due to the mechanism complexity and the forces needed to support a head. Consequently, the module is considered to be experimental.

The first version of the neck module, shown in figure 4.14, was created by Pieterjan Mollé as part of his master's thesis. The module consists of consists of two degrees of freedom, enabling pan (horizontal) and tilt (vertical) motion of the head. Both DOFs are realized using RC servos, facilitating the process of interfacing the module with the platform's electronics. A combination of laser cutting and 3D printing is used for the manufacture of the module's components. The module is attached to the frame of the robot using screws; we opted to forego snap connectors in this instance because the large forces would overcome the tension of the snap connectors.

The large loads caused by the weight of the head necessitate the use of metal bearings and reduction gearing on both axes. The tilt joint is supported by an M8 threaded rod with a ball bearing at each end. Tilt actuation is accomplished through a 3D printed screw mechanism. A four-start, large pitch trapezoidal screw is attached to the horn of the tilt servo. The screw is connected to the tilt mechanism through a 3D printed nut that mates with the threads of the screw. The servo mount and the nut are both supported by a revolute joint so as not to overconstrain the mechanism. The dimensions of the screw are carefully chosen so that the full range of motion of the servo ( $180^\circ$ ) results in  $\pm 25^\circ$  of tilt.



**Fig. 4.14** CAD model of 1<sup>st</sup> neck module

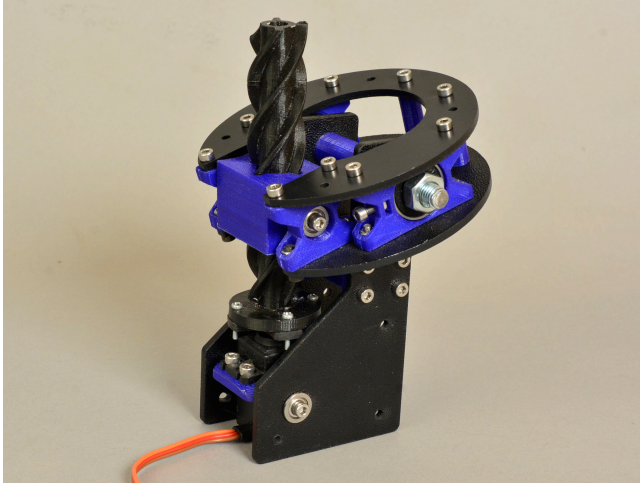


**Fig. 4.15** Prototype of 1<sup>st</sup> neck module

The panning motion of this prototype is achieved by rotating the entire tilt mechanism. An M8 threaded rod runs along the length of the tilt mechanism frame and is supported at the opposite end by two ball bearings that are attached to the lower end of the frame of the robot. Pan motion is accomplished through a toothed belt transmission, which connects a 3D printed pulley on the shaft of the servo to a pulley on the threaded rod.

The second version of the neck mechanism was designed by 2<sup>nd</sup> year students during the Illusion of Life course assignment, as described in section 3.5.2. During the course assignment, two groups required a moving head as part of their robot concepts. Version 1 of the neck module was used as a basis, though the module was redesigned due to a number of shortcomings. For the students, the main problem with the first neck module was its large size, making it hard to integrate the module into their embodiment design.

The redesign eliminates the panning motion, and focuses solely on tilting the head. The students had decided that this single degree of freedom would offer sufficient social expressiveness for their robot concepts, while simultaneously reducing size and complexity of the module dramatically. The screw mechanism was kept, though the surrounding structures were modified to increase rigidity, reduce the volume and number of 3D printed parts, and to reduce the total size of the module.



**Fig. 4.16** Prototype of 2<sup>nd</sup> neck module

Some important issues remain unsolved with this version of the neck module. The biggest problem is the direct result of the way RC servos work. The feedback mechanism is entirely contained within the housing of the servo. Position data is unidirectional, and is transferred from the servo controller to the servo, but not the other way around. More detail on the inner workings of RC servos can be found in section 4.2.4.

This control scheme works well for simple applications, such as those found in the RC hobby. In our situation, problems arise when the servo is first powered on. At startup, the controller has no knowledge of the current position of the neck, only the servo itself can measure this position. When the controller signals the servo to rotate the neck, the servo – having no knowledge of the mechanism it is attached to – attempts to move to this position as fast as possible. This results in large current spikes, shocking motion, and can potentially cause the robot to fall over due to angular momentum.

Potential solutions to this problem would require the actuator to be modified or replaced. For example, a standard servo can be modified to include a fourth wire connected directly to the wiper of the feedback potentiometer. This would allow the controller to measure the servo's position, and thus limit acceleration accordingly. Alternatively, one could replace the servo with a DC motor coupled to a rotary encoder. This approach requires additional circuitry, but would allow advanced features such as PID control and trapezoidal speed control. More complicated control schemes are also possible, such as series elastic actuation, though one would have to wonder whether this is worth the added cost

and complexity.

### 4.1.2 WORKSHOP BASE

The workshop base is not a core part of the system per se. As implied by the name, it is designed as a base unit for situations such as workshops, courses, and co-creation sessions, where it is important to get up-and-running quickly. The workshop base packages nearly all electric and electronic parts that are required for a robot into one unit. This includes two AC/DC power supplies, an AC plug, a Raspberry Pi with custom daughterboard, and a WiFi dongle. The unit lets the user focus on the physical embodiment design. The only wires that need to be connected are the leads of the servo motors.

Two versions of the workshop base were designed. The initial version, shown in figure 4.17, was created for use during our workshop at the TEI conference (section 3.5.1), and was also used during the first Illusion of Life course (section 3.5.2). The housing is comprised of a single plate of ABS plastic onto which all components are mounted. The components are protected by an acrylic cover plate mounted on top of them using threaded hexagonal standoffs.

The design worked well, though the elongated shape made the unit hard to integrate in some robot embodiments. A second version, shown in 4.18, was created for the second Illusion of Life course. The layout of the unit was changed to form a more dense, cuboid shape. In addition, a 5 W speaker was also integrated into the unit.

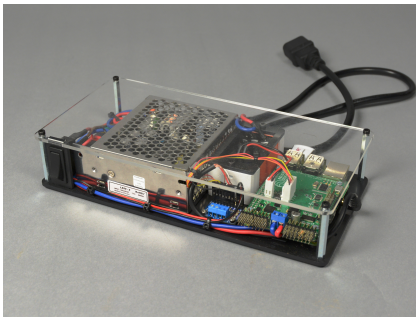


Fig. 4.17 1<sup>st</sup> workshop base

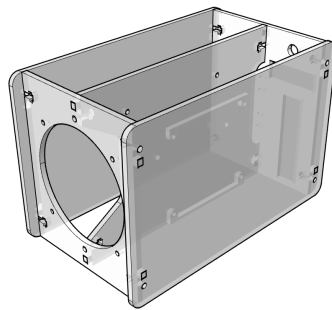


Fig. 4.18 2<sup>nd</sup> workshop base

### 4.1.3 EMBODIMENT

This section describes the process through which the standard components of the platform are combined with a custom skeletal frame and skin in order to form a functional social robot. As argued earlier, our design approach shares similarities with the concept of “untoolkits” (D. A. Mellis, Jacoby, et al., 2013) in the sense that our toolkit is not just a collection of modules, but that we see the step of creating a custom embodiment as an

intrinsic part of the kit. Thus, digital manufacturing techniques such as laser-cutting can be seen as intangible components of the Opsoro toolkit. A large degree of design freedom is afforded through this approach, as the design of the modules does not have a large influence on the aesthetics and design of the robotic character ( $G_5$ ). On the other hand, the step-by-step methodology offers guidelines for novices, stimulating them to be creative ( $G_7$ ).

The technical side of this process relies upon laser cutting. The design method encompasses both digital and analog, both high-fidelity and low-fidelity techniques. The next two sections detail the technical process of translating an embodiment concept into manufacturable designs, starting from a 3D model of the embodiment, and focusing on the design of the skeletal frame and the design of the skin. These two sections describe the *standard* process, which was applied for the implementation of the Ono robot. The third section describes a number of ways this standard process can be tweaked or extended, such as those used for the Illusion of Life course assignment (detailed description in section 3.5.2).

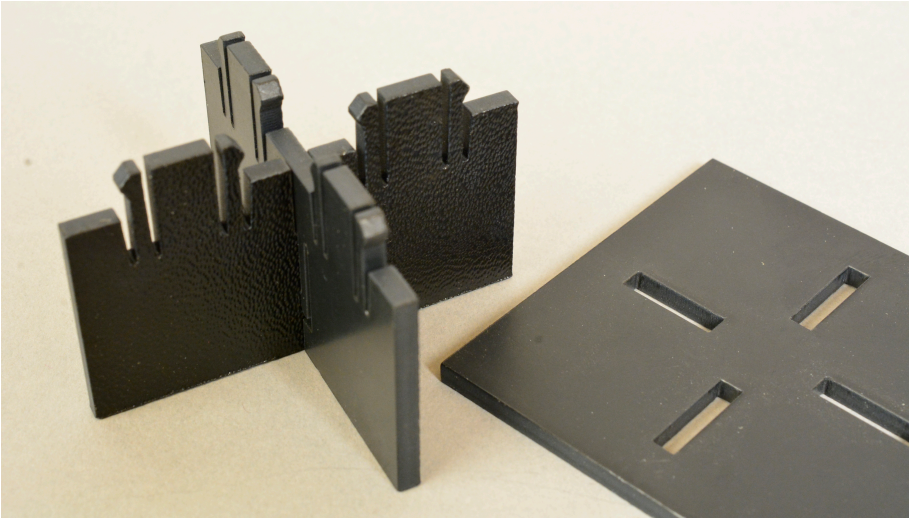
#### 4.1.3.1 FRAME

The frames are designed to be made from ABS plastic through use of a laser cutter. The flat laser-cut pieces interlock to form a rigid, three-dimensional structure. The design of the laser-cut parts incorporates various types of functionality through clever manipulation of the shape of the parts. For instance, the parts are designed so that they can be connected together without relying on adhesives, fasteners, or other external components.

The parts that make up the skeletal frame are connected using snap connectors, as shown in figure 4.19. The male part of the connection consists of two cantilevered hooks that rely on the elasticity of the material to restrain the mating part. Cutouts in the body of the connector extend the effective length of the cantilever, allowing the snap force to be fine-tuned. The female part of the connection (fig. 4.19 right) consists of a simple rectangular cutout.

3 mm thick ABS plastic sheet is used for the skeletal frames. The advantages of this material are its toughness, impact resistance, and relative inexpensiveness. In addition, it cuts very well on a laser cutter, with no burrs and very little discoloration. Unlike acrylic, which is commonly used for laser cutting, the material is not brittle, allowing the snap connectors to function without breaking. Delrin plastic would be another good candidate material for the frames due to its excellent toughness and wear characteristics. However, at approximately five times the price, the material is much more expensive than ABS.

The process starts with a 3D CAD model of the outer shape of the robot. The approach works for both polygonal 3D models as well as 3D surface models composed out of NURBS objects. However, we have always used surface models because they offer a higher precision and because they interact well with modern mechanical CAD software, which uses NURBS surfaces for the boundary representation (B-Rep) of solid objects.



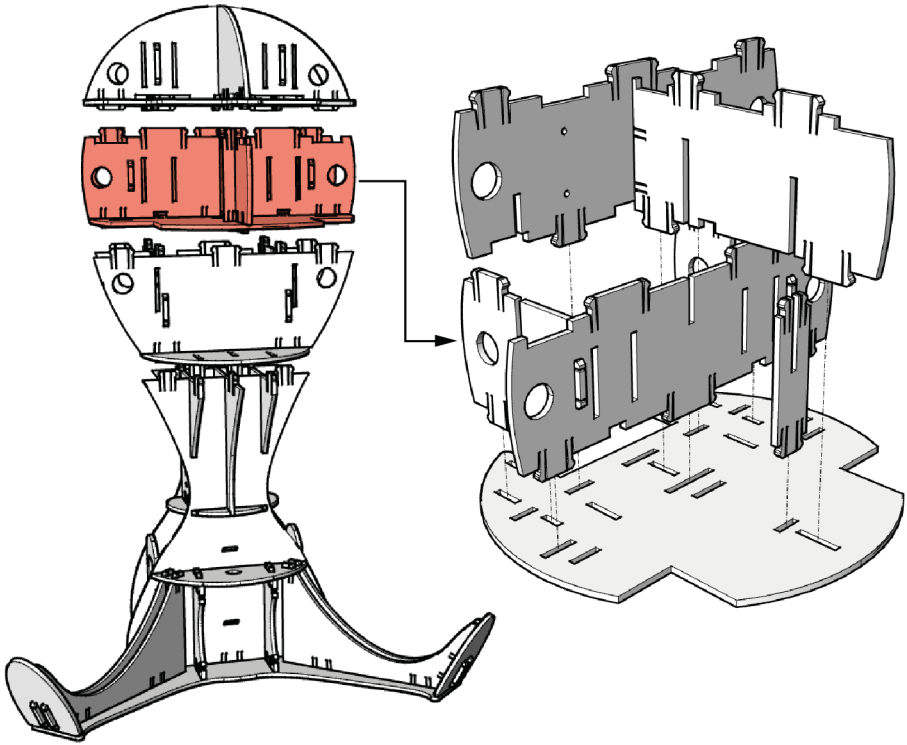
**Fig. 4.19** Example of the snap connectors used in the frame design

The first step is to offset the surface of the 3D model to the inside. The distance is dependent on the thickness of the foam that will be used to cover the skeletal frame at a later stage. 20 mm is a typical value for this. This new surface is imported into a mechanical CAD package as a reference object. Then, the various modules are imported and positioned within the 3D model. At this point, the location of the various “slices” is chosen. Section planes are then defined and intersection curves are calculated, which serve as a basis for the contours of the various parts of the skeleton.

The structure of the frame is divided into a number of sub-units that can be stacked on top of one another (fig. 4.20). Each sub-unit consists of a horizontal plate onto which a grid of interlocking vertical plates are attached. The intersection curves are used as a starting point to draw the individual parts that make up a sub-unit. The snap connectors, stored in a reusable sketch library, are added at this point. The width of the snap connectors can be changed. This can be used to introduce asymmetry into the parts of the skeleton, simplifying assembly by ensuring parts fit in only one specific orientation.

Once the basis of the skeletal frame is completed, final details can be added. First, the female connectors for the modules are added. Then, accommodations for the electrical systems are added, including holes for cable routing, zip-tie anchors for securing wires, and mounting holes for the Raspberry Pi and the power supplies. Finally, a part number annotation is added to each part, which will be etched into the part during laser cutting. This serves to let users to identify each unique part during assembly.

The process of designing a custom skeleton involves a large amount of CAD drawing work. However, the work is not difficult and consists of few steps that are repeated throughout the design. For this reason, we think that these steps are an ideal candidate to be automated through a specialized CAD system plugin.



**Fig. 4.20** Architecture of the frame of Ono, showing four sub-units

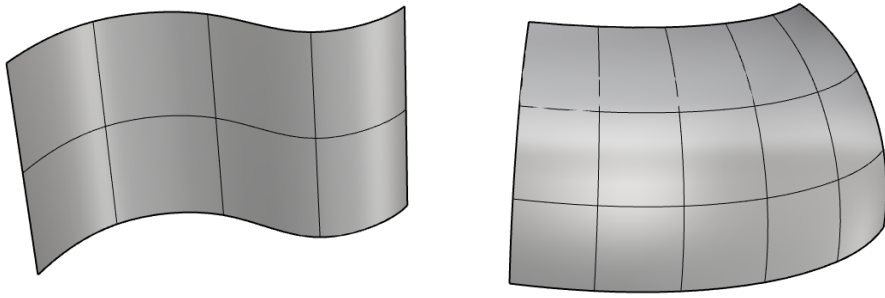
#### 4.1.3.2 SKIN

The robot's skin is typically made up of two layers: the inner layer is a layer of soft foam padding, the outer layer is a textile layer that gives the robot its visual appearance. Producing these two layers in a repeatable fashion can be challenging, especially since traditional techniques tend to involve a lot of manual labour. While our approach is not a radical departure from established techniques, we have incorporated a number of changes to simplify the process.

We found the established technique for the production of three-dimensional soft foam shapes, such as those required for the foam padding layer, to be a very difficult and involved process. Usually, these types of parts are created via molding and resin casting, especially when the geometry of the part is complex. The molding and casting process is a craft, requiring skill, time, and sometimes special equipment, such as a vacuum chamber. These aspects are worsened by the properties of soft PU foam resins, as they cure rapidly and are highly sensitive to temperature and air humidity. Casting can also be quite costly for one-offs, as one needs to invest materials such as silicone to build a mold. Casting resins also need to be bought in large volumes and tend to have a limited shelf life (a couple of months).



Our solution to this problem was to make the hollow 3D foam shape for the padding layer out of a patchwork of 2D pieces of foam. As with the design of the skeletal frame, software tools are used to facilitate this process of translating the 3D model to a pattern of 2D shapes. The 2D shapes are then cut out of a piece of flat foam sheet with the aid of a laser cutter. Finally, the 2D pieces are sewn together to form a three-dimensional object that approximates the shape of the original 3D model.



**Fig. 4.21** Developable surface (left) vs. doubly curved surface (right)

The biggest technical hurdle is transforming the doubly curved parts of the model into flat shapes. Doubly curved surfaces are surfaces that are curved in two distinct directions, such as the face of a sphere. As a result of this property, these surfaces cannot be flattened without distortion. This is in contrast with developable surfaces, such as the face of a cylinder, which can be flattened without distortion. Organic shapes, such as those found in character designs, have a lot of zones that are doubly curved.

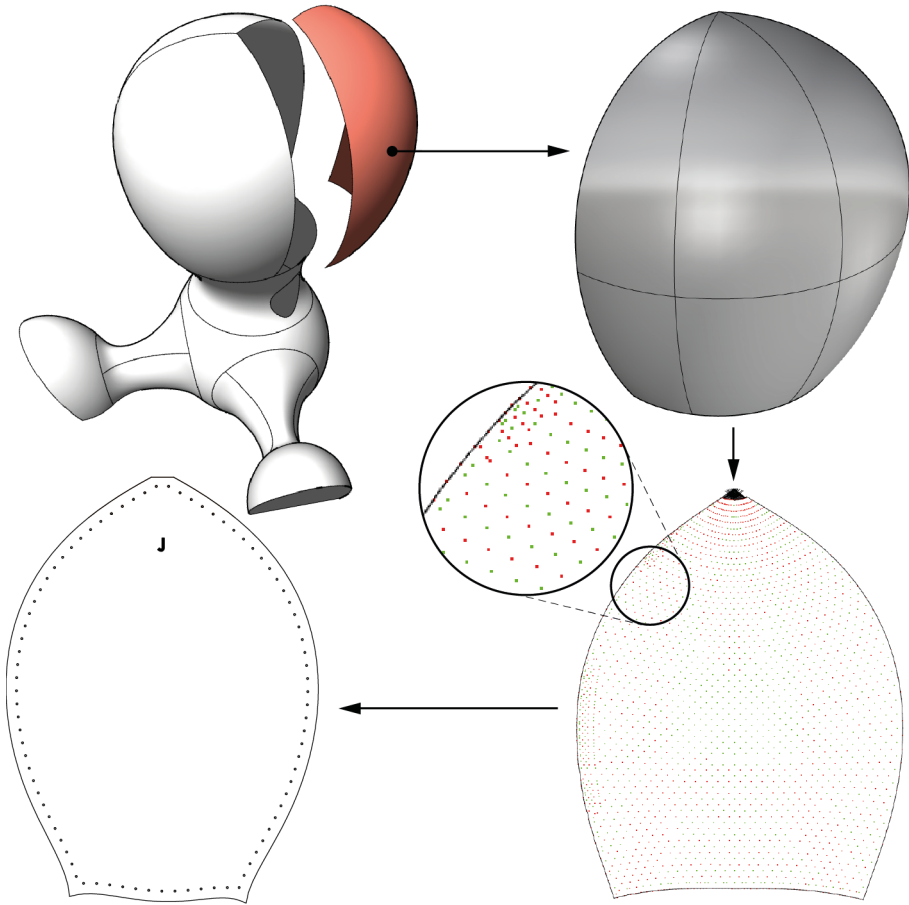
To generate the flat patterns for the foam, the model is first manually divided into separate regions. The designer should choose these regions so that they are large enough to limit the total number of pieces, yet small enough so that the 3D shape is within the limits of the foam's elasticity. These regions are then flattened using the *squish* command in Rhino3D<sup>1</sup>. This command flattens the doubly curved regions while minimizing the total amount of deformation.

Depending on the material, this process may be fine-tuned. The command can be given a bias to prefer stretching or compressing the material. The centerline of the deformation may be modified by offsetting the surface. We found that using the centerline surface (i.e. 10 mm offset for 20 mm foam) and a preference for compression gives the best results for the foam material we used. To finalize the patterns, artifacts on 2D contours are removed and a part number is added. Then, a pattern of equidistant holes is drawn along the edge of the parts. This stitching pattern makes it easier to manually sew the foam around the robot at a later stage.

The design process for the fabric skin layer is very similar to that of the foam padding layer. The same *squish* tool is used, though the surfaces are not offset, and the tool's bias is set

<sup>1</sup> Rhinoceros 3D is a CAD application focused on NURBS modeling. – <http://www.rhino3d.com>





**Fig. 4.22** Foam flattening process. The colored dots indicate material deformation. Note the artifacts at the top corner of the generated contour.

to prefer stretch deformation over compression. We used Lycra fabric for the skin of the Ono robot. The high elasticity of the fabric is particularly useful because it smooths out any irregularities of the foam layer underneath.

#### 4.1.3.3 VARIANTS

The paragraphs above describe in broad strokes the embodiment design process as we had originally envisioned it. One advantage of the approach lies in its flexibility. The steps of the process can be changed whenever changes are needed or desired. Adaptations can be made to accommodate for different skill levels ( $G_6$ ), longer or shorter timescales, available resources and infrastructure, etc. Below are some examples we have encountered over the course of this research project:

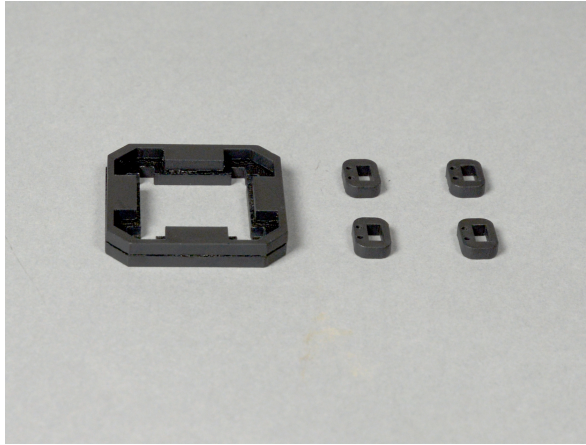
- During our workshop on DIY animatronic creatures at the TEI conference (section 3.5.1), we had a very limited timeframe to build new robots with participants. The duration workshop was limited to roughly six hours, during which participants would go from a rough concept to a working animatronic creature. Consequently, we eschewed the use of computer-based design tools in favor of manual, hands-on techniques. We provided pre-assembled modules, which were brought to life using the custom cardboard embodiments that the participants built. Plastic connector plates (fig. 4.23) were used to interface the modules with the cardboard. These were glued to the cardboard, allowing the modules to snap in place afterwards.
- For the Illusion of Life course assignment (section 3.5.2), we recommended the students to use Autodesk's maker-friendly apps for the frame design of their robots. These tools are easier to use and produce results more quickly than traditional CAD. More specifically, they used MeshMixer<sup>2</sup> to digitally sculpt the outer shape of the embodiment. The resulting triangle mesh was then imported into 123D Make<sup>3</sup>, which was used to slice the model into interlocking laser-cut pieces. The resulting skeleton was used as the first design iteration, after which the files were imported into a "real" CAD package to make any modification required for the final design.
- In the same course assignment, students experimented with very different materials and techniques to create a skin for their robots. Originally, we had focused on creating soft, huggable robots that children could safely touch, resulting in a method that relies on textiles, foams, and other soft materials. However, during the course, many different modes of interaction were conceptualized. For instance, in one concept, the robot serves as a tutor for children. Consequently, a more distant mode of interaction is appropriate because the robot and the child are not social peers, and this has consequences for the design language of the embodiment. Examples of novel skinning techniques include hard-shell covers made from thermoformed PS and semi-rigid foam skins made from thermoformed EVA foam, giving way to different sensory experiences.

## 4.2 ELECTRONICS

Keeping in line with the overarching philosophy of this work, the electronics used during this project initially relied heavily on off-the-shelf modules and boards. As the platform matured and as our views on what functionality we deemed essential evolved, so did our implementation of the platform's electronics. As time progressed, we have gone from breadboard circuits, to custom etched PCBs carrying off-the-shelf modules, to fully integrated boards. Still, throughout this process, we have made a conscious effort to build upon existing platforms, maintaining a close link with contemporary DIY, hacking, and making paradigms ( $G_{10}$ ). As such, the latest incarnation of the electronics is conceived as

<sup>2</sup>Autodesk MeshMixer – <http://www.meshmixer.com>

<sup>3</sup>Autodesk 123D Make – <http://www.123dapp.com/make>



**Fig. 4.23** Laser-cut snap sockets. These consumables were glued into cardboard skeletons to provide module mounting locations.

a daughter board for the Raspberry Pi SBC, containing (among others) a microcontroller that can be programmed using tools from the Arduino ecosystem.

## 4.2.1 FIRST GENERATION – MICROCONTROLLER-BASED

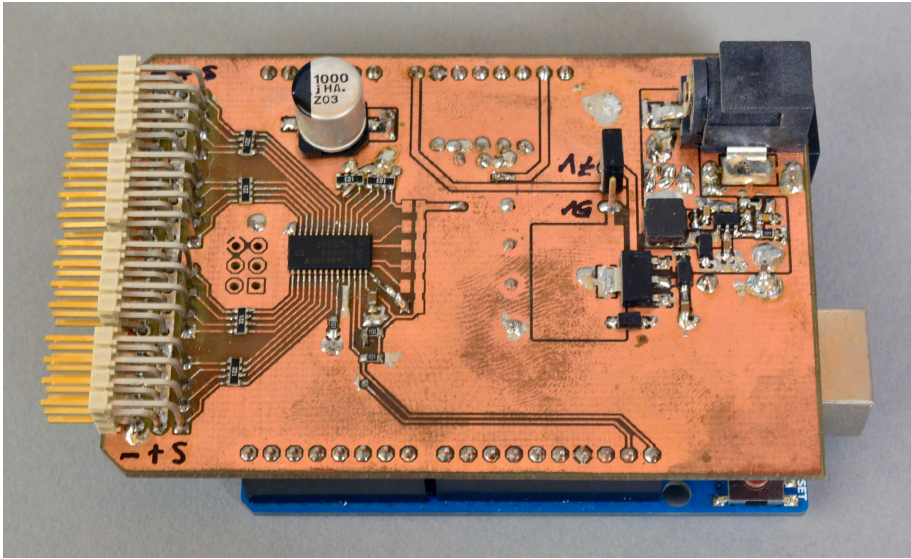
The first generation of electronics was originally designed to be used as a test-bed for the platform's modules. For that reason, functionality was mostly limited to providing sufficient PWM outputs in order to allow for enough servo actuators to animate a full face. The system's only input is a joystick interface, the x/y position of which is directly mapped to *valence* and *arousal* parameters. These two parameters are then used to determine the position of each degree of freedom.

Considering the straightforwardness of these requirements, the first generation of electronics was implemented directly using a microcontroller. The facial expression algorithm itself is not very taxing for the processor, the only challenging facet is generating a PWM signal for each Degree of Freedom, 13 in total. As such, the joystick input processing and the facial expression algorithm were implemented on an Arduino Uno, with the PWM signal generation being handled by an external module.

The first iteration used a SSC-32 board<sup>4</sup> to drive the servos, which was integrated into the body of the robot. The power supply, the microcontroller board, and the joystick interface were housed in a separate, tethered control box. A picture of the control box is shown in figure 4.39.

<sup>4</sup>Lynxmotion - SSC-32 Servo Controller –  
<http://www.lynxmotion.com/p-395-ssc-32-servo-controller.aspx>

From the second version onward, a dedicated Integrated Circuit (IC) was used instead of the SSC-32 to generate the PWM signals. The IC – a PCA9685 – is intended to be used as a LED driver by the manufacturer. However, the chip’s 12-bit PWM resolution and programmable frequency mean that the IC can be repurposed to generate servo control signals. Technical details on the implementation of this idea can be found in subsection 4.2.4.



**Fig. 4.24** Picture of the Arduino shield

This iteration, shown in fig. 4.24, was conceived as a shield (daughterboard for Arduino boards), with both Arduino and shield intended to be integrated into the body of the robot. The rationale of this approach is twofold. First of all, Arduino boards are readily available worldwide, it makes little sense to duplicate this functionality in a new board. Secondly, the success of Arduino ecosystem meant that the board’s pinout has become a pseudo-standard; many different microcontroller and microcomputer boards with an Arduino-compatible pinout are available. With this in mind, we envisioned a system where a user could choose the logic board that is best suited for their application. By attaching the shield, the logic board would be capable of controlling the robot. For instance, for most basic applications, a low-cost Arduino Uno board would suffice. However, more demanding users could switch to Arduino-compatible microcomputer boards, such as the Arduino Yun or the Intel Edison.

## 4.2.2 SECOND GENERATION – MICROCOMPUTER-BASED

In retrospect, the approach described in section 4.2.1 showed a number of shortcomings. First of all, different processor architectures ensue that duplicate versions of the same software need to be written, as each platform has its own APIs and its own methods of

solving problems. At this stage of the project, the size of the software codebase was still limited. However, this approach is not sustainable as the scope and functionality of the software grows. As a second factor, we found that the things we wanted to add nearly always required functionality that exceeds the capabilities of most simple microcontrollers, for example audio playback or communication over Wi-Fi. As such, using a SBC started to make much more sense. Another alternative is to use an off-the-shelf smartphone as the brains of a robot. Smartphones have an abundance of computing power and already offer many features that are useful for social robots, including camera, microphone, speaker, and wireless communication. Indeed, multiple social robots, including Tega (Westlund et al., 2016) and Travis (Hoffman, 2012), use this approach. However, we decided again smartphones in favor of using a Raspberry Pi SBC for a number of reasons: (1) by integrating a smartphone inside a robot, the device loses most of its affordances as a user interface, (2) factoring in the required additional power, actuator and sensor circuitry, both solutions have a very similar cost, and (3) there is a large degree of variation in smartphones due to different models, different OSs and different software versions.

#### 4.2.2.1 HAT REV. 0 & 1 – DIRECT I2C

In light of above-mentioned factors, we opted to move forward with the Raspberry Pi as underlying electronics platform. For us, the Raspberry Pi offered a number of advantages over alternative SBCs, with the BeagleBone in particular. At the point in time where we started developing this generation of electronics, there was a significant price difference between the platforms, with the original BeagleBone costing nearly twice as much as the Raspberry Pi model B. As reducing the monetary cost of social robotics is one of the main design challenges of this project ( $G_9$ ), this influenced our decision.

A second benefit of the Raspberry Pi ecosystem is the availability of a dedicated Camera Serial Interface (CSI) port. With this port, a camera can be added to the system, for example for computer vision or for remote monitoring purposes. While one could also use a USB webcam, a performance penalty would be incurred. In the case of the Raspberry Pi, the processor's only USB bus is connected via an internal USB hub/ethernet controller to the ethernet connector and the four USB connectors. In practice, the sum of data through these connectors is limited to the bandwidth of a single USB bus, with the USB camera taking up a significant portion of available bandwidth. In comparison with the Raspberry Pi, the BeagleBone only offers a single USB host port, and while one could use an external hub to connect more devices, this approach would suffer from similar problems to the Raspberry Pi's.

A daughterboard for the Raspberry Pi, called a Hardware Attached on Top (HAT) board in Pi jargon, was designed. The board contained circuitry to support 16 servo channels and 12 capacitive touch channels, along with the necessary power regulation. The board, shown in 4.25 was designed as a single layer photo-etched PCB, onto which off-the-shelf modules are soldered. The PWM controller for the servos and the capacitive touch controller communicate with the Raspberry Pi over the Inter-Integrated Circuit (I<sup>2</sup>C) bus. This low-level bus allows up to 112 distinct devices to share the same bus, with each de-

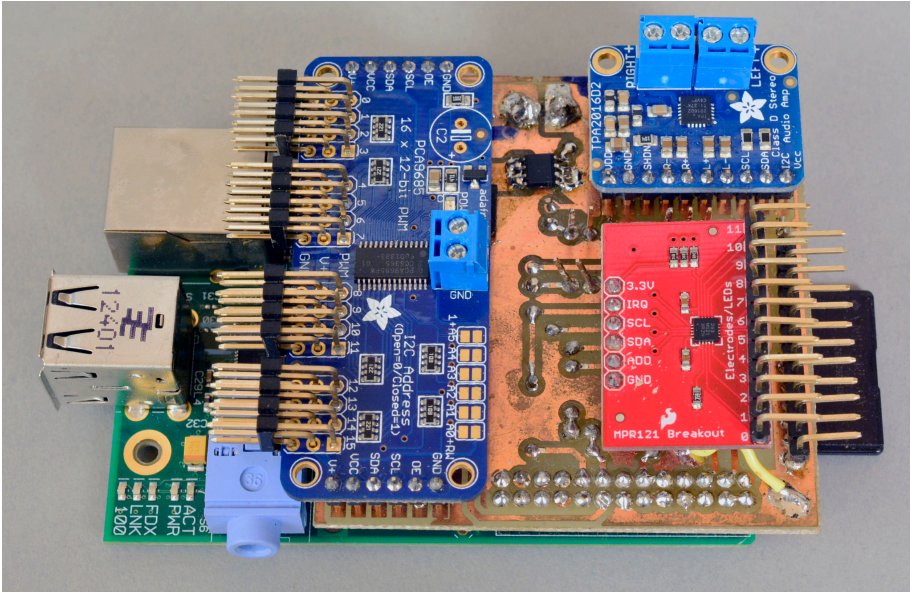


Fig. 4.25 Picture of HAT rev. 0

vice having a unique 7-bit address. The bandwidth offered by the I<sup>2</sup>C bus is fairly low, typically around 400 kbit/s in fast-mode, though still more than adequate for this application. The main advantage of this approach is that timing-critical tasks (e.g. generating accurate PWM signals) are executed by the I<sup>2</sup>C devices, and not by the main processor. The Broadcom System-on-Chip (SoC) on the Raspberry Pi runs a full linux OS, and is consequently not suitable for real-time tasks.

The system is powered by two independent power supplies, one for logic and another one for the actuators. The power to the actuators can be switched on/off using an N-channel MOSFET. High-current, inductive loads can cause many problems in digital systems. The sudden current draw that occurs when the servos are turned on can cause the power supply's voltage to momentarily drop below the processor's minimum operating voltage, causing it to reset. This condition is called a brownout reset. In contrast, when servos are disabled, high-voltage transients occur that can damage sensitive devices such as Field-Effect Transistors (FETs). A third problem is that motors and other high-current devices cause electrical noise, which can be especially problematic for analog circuitry. These problems can be mitigated through adequate filtering, e.g. using a Schottky diode combined with large electrolytic and ceramic capacitors. However, the simplest solution to all these issues is to use separate power supplies for logic and motors, which is how the system is implemented in the Opsoro platform. As an aside, the cost of both solutions is approximately the same, though the filter solution involves more assembly steps.



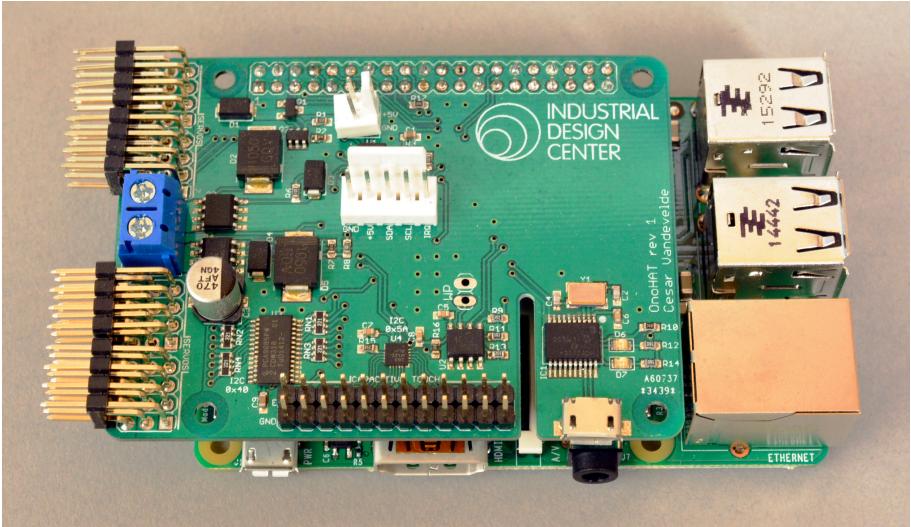


Fig. 4.26 Picture of HAT rev. 1

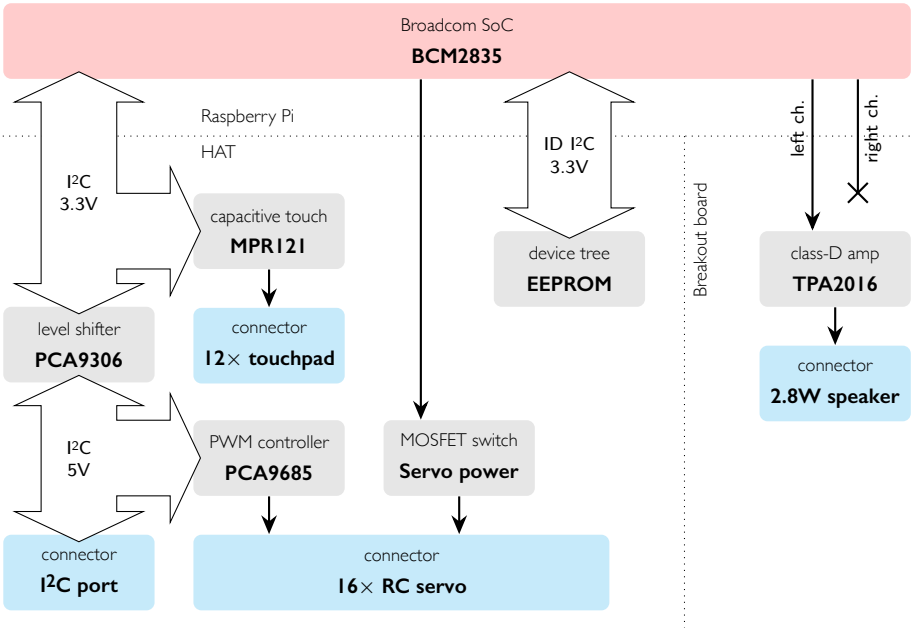


Fig. 4.27 System diagram of HAT rev. 1

The second version of the platform was designed as a proper, integrated PCB. A block diagram of the system is given in fig. 4.27. Little changed in way of functionality in this revision, however the form factor was completely redesigned in order to conform to the then newly-released HAT specification, which coincided with the release of the Raspberry Pi B+. The main difference is that this version was designed for Surface-Mount Technology (SMT) instead of relying on through-hole components combined with off-the-shelf modules. The resulting board is much more compact and could be manufactured using automated pick-and-place machines. Another point of difference is the protection circuitry: this board was made more rugged by adding features such as Transient Voltage Suppression (TVS) diodes, series gate resistors, and reverse-polarity protection using FETs.

While revision 1 of the HAT worked well at first, continuous use revealed a problem where the I<sup>2</sup>C bus would eventually hang. This problem was erratic and proved hard to track down. As it turns out, the problem is caused by the implementation of the I<sup>2</sup>C protocol in the Raspberry Pi's Broadcom SoC. I<sup>2</sup>C standards dictate that a slave device may hold down the bus's clock line if it needs more time to process a request. This action is called clock stretching. Due to the SoC's implementation, a very short clock pulse can occur immediately after clock stretching, which may be too short to be detected by the slave device. In turn, this causes a bit order mismatch between master and slave, causing the bus to hang. It should be noted that this bug is caused by the SoC's silicon design, it is unfortunately not a software problem. While there are software workarounds for this problem, such as lowering the bus clock speed or periodically resetting the bus, none of them were deemed to be suitable solutions for long-term use.

#### 4.2.2.2 HAT REV. 2 – SPI COMMUNICATION

A second revision of the HAT was created in order to solve this problem. Our approach involved switching the communication between the Raspberry Pi and the HAT from the I<sup>2</sup>C bus to the Serial Peripheral Interface (SPI) bus. This bus uses a separate signal for clock (CLK), Chip Select (CS) (used for addressing), Master-Out Slave-In (MOSI) data, and Master-In Slave-Out (MISO) data. Consequently, the SPI bus offers a much higher potential throughput than I<sup>2</sup>C. More importantly, it uses a different device addressing scheme and does not suffer from the same problems as the I<sup>2</sup>C bus.

The ICs that provide PWM and capacitive touch functionality can only be interfaced using I<sup>2</sup>C, with no immediate replacement devices that offer the same functionality through a SPI interface. We chose to solve this issue by using a microcontroller (an ATmega328; the microcontroller used in the Arduino Uno) to control peripheral devices. An updated system diagram is given in fig. 4.29 (note the differences to fig. 4.27). The flexibility of this approach means that new functionality can be implemented which does not exist as off-the-shelf SPI or I<sup>2</sup>C devices, e.g. driving addressable RGB LEDs. On the other hand, this approach is not without downsides. The microcontroller itself is an added cost and takes up valuable board space on the PCB. This device also needs to be programmed with a firmware, adding an extra step and increasing system complexity.



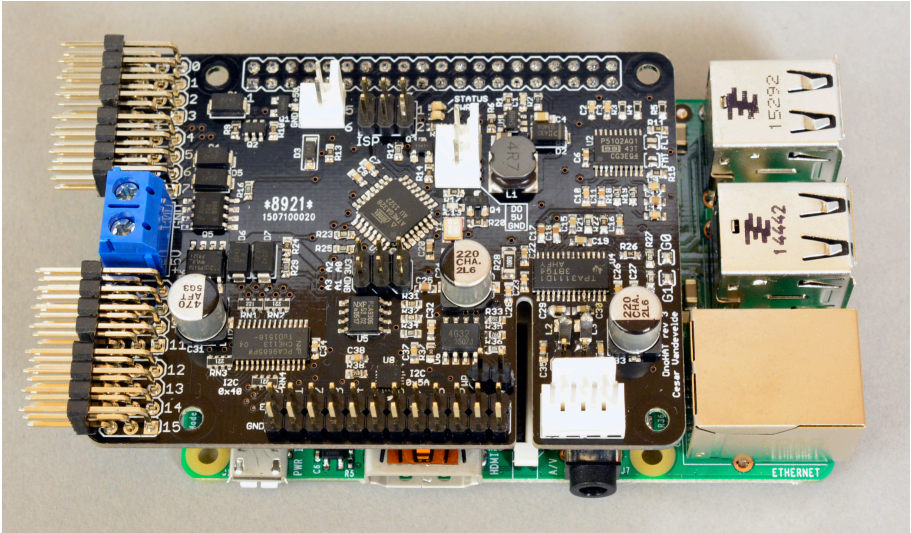


Fig. 4.28 Picture of HAT rev. 2

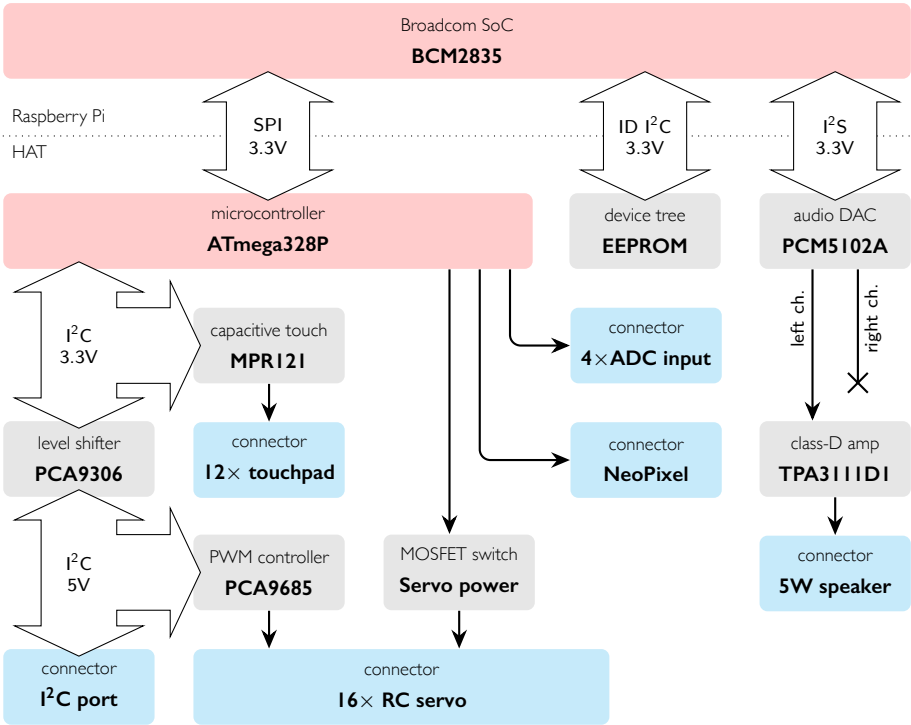


Fig. 4.29 System diagram of HAT rev. 2

For the implementation of this system, a protocol had to be designed on top of SPI to transfer commands and data between master and slave. The Broadcom SoC is capable of SPI clock speeds up to 125 MHz, several orders of magnitude faster than what the 8-bit microcontroller would be able to handle. Instead the bus clock speed was set to a more manageable frequency of 122 kHz. A typical transaction between master and slave happens in two phases. The first step (fig. 4.30) is to transfer a command plus its parameters, if any to the slave. Whenever the CS pin is pulled low by the master, the slave will interpret the next MOSI byte as a command byte, with each consecutive byte stored in buffer as parameter data. When the CS pin is driven to a logic high, the microcontroller processes and executes the command, during which the master waits. Results of commands, if any, are stored in an outgoing buffer. The second step (fig. 4.31) is to read the output of the command. This step is optional and only occurs if the master is expecting return data. The CS pin is pulled to logic low and a special CMD\_READ byte is transferred to the slave. Each subsequent byte sent by the master causes the slave to return the next byte of the outgoing buffer.

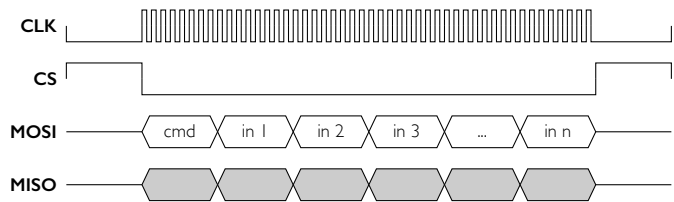


Fig. 4.30 SPI write protocol

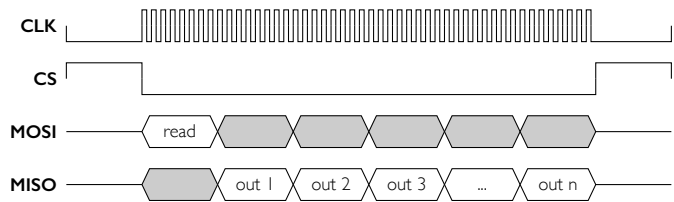


Fig. 4.31 SPI read protocol

The protocol is implemented in firmware using two Interrupt Service Routines (ISRs), one to handle incoming SPI data, and second one to handle state changes of the CS signal. When the CS pin goes low (meaning that the slave is being activated), the pin-change ISR sets a `isCmd` flag indicating that the next SPI byte will be a command byte. Conversely, when CS goes high, a `processCmd` flag is set indicating that data transfer is finished and that command processing can begin. The main loop continuously polls the `processCmd` flag and executes the commands. The SPI ISR functions to move parameter data to an incoming buffer and to move result data from an outgoing buffer to the bus.

In addition to the system architecture change, revision 2 of the HAT also offers a number of new features. The first of which is the redesigned audio subsystem. Sound and speech is an important aspect in many social interaction contexts. As such, this functionality was given extra attention. From an implementation perspective, sound output typically has

two distinct steps. As a first step, sound is manipulated digitally (e.g. decoding an MP3 file or synthesizing speech) and is output as an analog signal through a Digital to Analog Converter (DAC) circuit. In a second step, this signal is sufficiently amplified so that it can be used drive a speaker.

By default, the Raspberry Pi outputs sound through a PWM output that is filtered through a resistor-capacitor network. This output is connected to a 3.5mm headphone connector. While this signal, which comes directly from the SoC, is sufficiently powerful to drive headphones, it cannot be used directly to power a larger speaker. In prior revisions of the platform, the headphone signal is amplified through a small class-D audio amplifier breakout board, which in turn is connected to a speaker. This approach has a number of limitations. The PWM-based DAC circuit – while simple and inexpensive – introduces a considerable amount of harmonic distortion into the audio signal. This results in noise, even when the system is not playing any sound. The audio output is also only available through the 3.5mm connector, and not through the 40-pin HAT connector. Thus a separate cable needs to be used, a bulky and more expensive solution. Finally, the commercial amplifier board we used was limited in power output, resulting in a lower volume than desired.

Our solution was to include a high-quality Inter-IC Sound (I<sup>2</sup>S) DAC, along with a better-suited class-D amplifier in the HAT board design. The I<sup>2</sup>S bus is a digital bus designed to transfer sound data between ICs. The signals of this bus are available on the Raspberry Pi's 40-pin connector, thus eliminating the need for an external cable between the two PCBs. In addition to this, the external DAC – a PCM5102A – has a very low total harmonic distortion, resulting in an audio signal of much higher quality than the Raspberry Pi's onboard audio circuitry. Finally, this custom solution allowed us to choose a class-D amplifier suited to drive the speaker at optimal levels, increasing maximum power output from 2.8 W to 5 W. To achieve this, a boost converter was needed to step the 5 V system voltage up to 8 V, which is fed into the amplifier.

Finally, the necessity to include a microcontroller into the design in order to work around the I<sup>2</sup>C issues afforded an opportunity to offer additional board functionality without adding significant component costs. The microcontroller has four unused 10-bit analog input pins. These pins – along with two power pins – were broken out to a connector, allowing users to easily interface with analog sensors, such as Force-Sensitive Resistors (FSRs), flex sensors, or potentiometers. Another feature which was implemented this way is the ability to drive NeoPixels directly from the HAT. NeoPixels are a brand of addressable RGB LEDs, available in a number of different form factors, such as rings, matrices, strips, and single LEDs. They require only a single data pin to transfer RGB data and the LEDs can be daisy-chained together to allow for a large number of LEDs to be driven from a single pin. However, the serial protocol's strict timing requirements can pose a problem for embedded linux systems. The hardware required to implement this feature is fairly minimal, comprising a connector and a transistor-based level-shifting circuit to step the data signal up to a logic level of 5 V.

### 4.2.3 SENSING TOUCH

The design of the HAT PCB includes support for capacitive touch sensing, enabling users to design robots that respond to human touch. The technology is based around the idea that human tissue is slightly electrically conductive. When a hand comes near an electrical conductor, the two form a very small capacitor. The capacitance of this system depends (among other factors) on the distance between the hand and the electrode. Consequently, a circuit can detect human touch by measuring the change in capacitance of the electrode.

The capacitive touch sensor functionality offers a simple and low-cost method for adding basic sensing to a robot. The complete system is made up out of two parts: the capacitance measurement circuitry, which is incorporated into the HAT, and the electrodes, conductive areas that are positioned under the skin. The circuitry costs less than 10 €, and supports up to twelve electrodes. In contrast, FSR touch sensors cost approximately 8 € each, and TakkTile sensors (Tenzer et al., 2014) cost \$149 each.

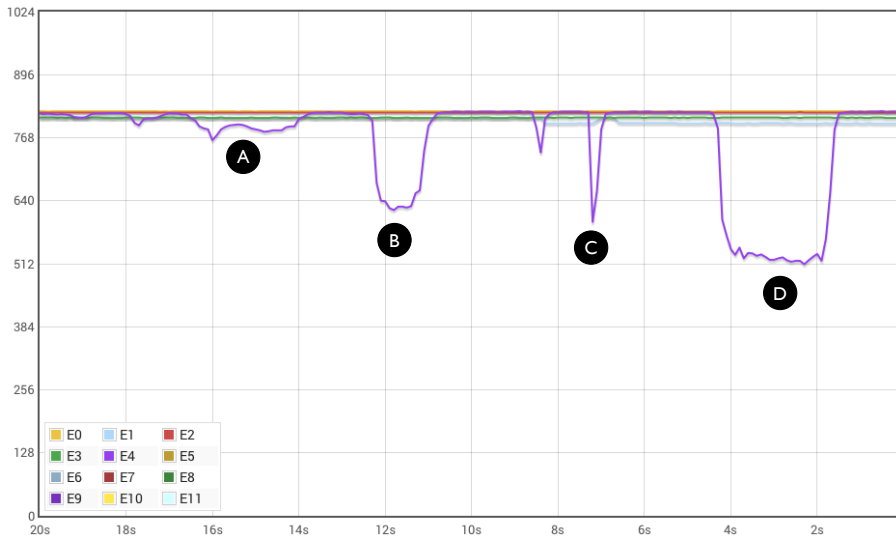
The electrodes consist of nothing more than a connector, a shielded cable, and a conductive pad. We have successfully built sensors using many different materials, including aluminium foil, copper-clad laminates, copper foil tape, conductive ink, and conductive textiles. The biggest challenge is usually attaching a cable to the conductive pad. Soldering can be used for copper-based materials. Other materials are usually attached mechanically; using staples, clamps, crimps, or sewing.

Because sensors are so easy to construct, the technology is well-suited for DIY sensor design, allowing customized sensors for each robot. The technology has previously already been used as an expressive medium in the domain of human-computer interaction toolkits. One of the most prominent examples is the Makey Makey (J. Silver et al., 2012), a USB device that lets users attach conductive objects to their computer to act as keyboard keys, opening the door to new interfaces such as the banana piano and the Play-Doh PacMan controller. Another example is given by D. A. Mellis, Jacoby, et al. (2013). Here, paper, conductive inks, and inexpensive components are combined to form an *untoolkit* for papercraft circuits. One of the default firmwares of their tool uses capacitive touch sensing to control LEDs. Finally, Touché (Sato et al., 2012) uses a swept frequency measurement technique to detect gestures and grasps, allowing objects respond differently depending on the way they are touched.

One of the challenges of letting the toolkit users design their own electrodes is that minimum and maximum capacitance range can vary greatly, depending on design parameters such as the shape, size, and material of the electrode. Equation 4.1 shows the formula for the capacitance  $C$  of a parallel plate capacitor, which approximates the behavior of capacitive touch electrodes. The formula shows that the capacitance of an electrode is proportionate to its area  $A$  and inversely related to the distance  $d$  between electrode and hand. Consequently, larger electrodes need a larger measurement range, and electrodes that are embedded under different materials need a smaller, more precise measurement range.

$$C = \varepsilon \frac{A}{d} \quad (4.1)$$

The capacitive touch IC that was chosen for the design of the HAT, the MPR121, handles this issue by offering a per-channel programmable charge current to account for both small and large electrodes. The IC also offers an auto-configuration mode to automatically determine the appropriate charge current and charge time for each electrode. This process is transparent to the users of the platform. At the start of an app that uses touch sensing, the IC spends a fraction of a second in auto-calibration mode, after which all electrodes can be used without further configuration by the user.



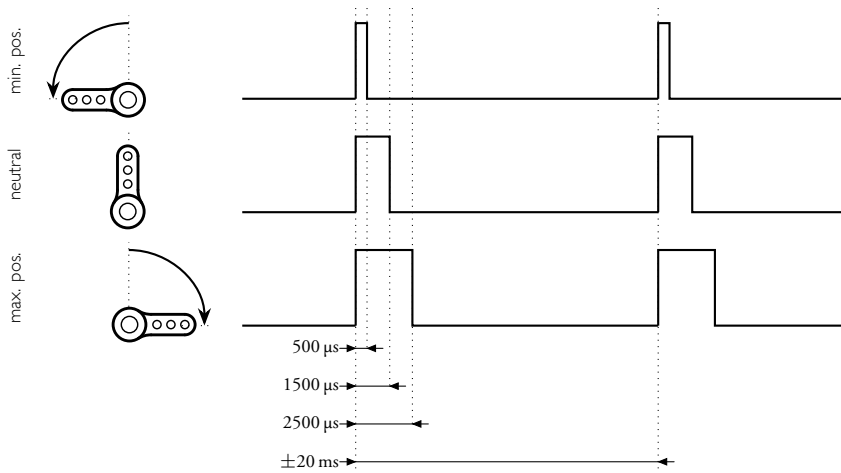
**Fig. 4.32** Capacitive touch sensor data. The signal shows (A) proximity, (B) soft touching, (C) hitting, and (D) squeezing.

In many applications of capacitive touch sensing – for instance, the interface of a kitchen appliance – electrodes are used as simple on/off buttons. However, in reality, the system is capable of much more granular measurements. For instance, with the sensors integrated into the skin of Ono, we can discern the difference between no touch, proximity (a couple of centimetres), soft touch, and different gradations in pressure. An example of sensor data is given in fig. 4.32. More so, by incorporating time and by combining data from multiple sensors, it should be possible to perform even more advanced sensing techniques, allowing the robot to detect different grasps and be able to distinguish between petting and hitting.

## 4.2.4 CONTROLLING SERVOS

As glanced at in earlier sections, all but the earliest iteration of the platform electronics use the same method for generating the control signals for hobby servos. As these servos are a key component in the design of the platform, this aspect deserves some elaboration.

The servos used throughout this work are off-the-shelf servos originally designed for use in RC models. They contain a DC motor, a gearbox, a feedback potentiometer attached to the gearbox output, and a control PCB in a small, self-contained package. In this scheme, the control PCB will drive the DC motor so that it moves toward the desired position, using the potentiometer to determine the current position. Generally, these servos are limited to 180° range of motion.



**Fig. 4.33** Servo PWM timing

Hobby servos are connected using a three-wire cable. Two pins provide power, typically at 5 V, though servos with a higher operating voltage exist. The third pin is a control signal to set the servo's position. PWM is used to encode the desired shaft position. The width of the pulse determines the output position, as shown in fig. 4.33. Pulse widths vary between 500  $\mu$ s and 2500  $\mu$ s, with a value of 1500  $\mu$ s representing the servo's neutral (middle) position. These pulses need to be sent at an update rate of roughly 50 Hz, which corresponds to a period of 20 ms. However, unlike the pulse width, the signal's frequency is not critical. The signal's period can be as short as 10 ms (100 Hz) or as long as 30 ms (33.3 Hz), with little bearing on the functioning of the servo.

To control a robot, many of such PWM signals are necessary, one for each DOF. Generally, microcontrollers do not have enough PWM-capable pins to drive a full robot. As a solution, we chose to use a PCA9685 IC, a 16-channel, 12-bit PWM LED driver. While this device is not designed for the purpose of controlling hobby servo's, it possesses a number of characteristics which make it suitable for this purpose. First of, the IC's frequency can be programmed from 24 Hz to 1526 Hz. Secondly, the 12-bit PWM resolution offers

enough intermediate steps between pulse widths of  $500\ \mu\text{s}$  and  $2500\ \mu\text{s}$ . A final benefit is that the device is free-running, meaning that once it is configured, the microcontroller is free to do other things.

The PCA9685's 12-bit PWM resolution results in 4096 distinct pulse width positions. However, this pertains to the complete length of the period. Suppose the device's frequency is set to 50 Hz. The corresponding period would be  $1000\ \text{ms}/50\ \text{Hz} = 20\ \text{ms}$ . Yet, as we are generating servo signals, we only care about pulse widths between  $500\ \mu\text{s} = 0.5\ \text{ms}$  and  $2500\ \mu\text{s} = 2.5\ \text{ms}$ . The minimum servo position results in  $\frac{0.5\ \text{ms}}{20\ \text{ms}} \cdot 4096 = 102.4 \approx 102$ , the maximum position results in  $\frac{2.5\ \text{ms}}{20\ \text{ms}} \cdot 4096 = 512$ . This results in 410 valid servo positions, much less than the device's 12-bit PWM resolution.

For the actual implementation, the PCA9685's frequency is chosen more deliberate than described above. The frequency of the device is set using a clock prescaler, which limits the frequency to discrete steps. The formula for calculating the prescale value  $P$  for a given frequency  $f$  is given in equation 4.2<sup>5</sup>.

$$P = \text{round}\left(\frac{25\ \text{MHz}}{4096 \cdot f}\right) - 1 \quad (4.2)$$

To simplify software computation, we choose 1 PWM step to represent  $4\ \mu\text{s}$ . Conveniently, the intended range of motion is  $2500\ \mu\text{s} - 500\ \mu\text{s} = 2000\ \mu\text{s}$ , which is divisible by 4. A benefit of this is that the pulse width to PWM calculation can be implemented using a bit shift operation instead of division (which is a slower operation). Working backward from this step size gives us a frequency of 61.04 Hz, as shown in equation 4.3. This frequency is well within the range of acceptable update rates for servos, as described earlier.

$$f = \frac{1}{T} = \frac{1}{4096 \cdot 4\ \mu\text{s}} \approx 61.04\ \text{Hz} \quad (4.3)$$

Entering this value into equation 4.2 results in a prescaler value of 99. With a resolution of  $4\ \mu\text{s}$  per step, a theoretical angular resolution of  $\frac{4\ \mu\text{s}}{2000\ \mu\text{s}} \cdot 180^\circ = 0.36^\circ$  can be achieved, which is sufficient for this application. As an aside, the system's total accuracy is influenced by many other electrical and mechanical factors, including signal noise, type of servo (digital servos tend to be more accurate than the analog type), servo range, and gearbox backlash. In conclusion, the servo positioning will be less accurate than this theoretical prediction, and is the result of many different environmental parameters.

<sup>5</sup>PCA9685: 16-channel, 12-bit PWM Fm+ I2C-bus LED controller. – [http://cache.nxp.com/documents/data\\_sheet/PCA9685.pdf](http://cache.nxp.com/documents/data_sheet/PCA9685.pdf)

## 4.3 SOFTWARE

### 4.3.1 FACIAL EXPRESSION ALGORITHM

One of the requirements for a facial expression algorithm is a way to represent emotion ( $G_3$ ) in a way that computers can deal with (i.e. numbers). Many models of affect find their origin in the field of psychology. Of course, psychologists and roboticists have different goals in mind for affective models. In psychology, a model is chosen to attempt to understand and explain human behavior. In social robotics, affective models are used in the reverse direction: they are used to synthesize rather than analyze emotion.

Fong et al. (2003) discern three main groups of affective models. The first group describes emotion in terms of discrete categories. The work of Ekman (1992; 1999), which discerns *happiness, sadness, anger, fear, surprise* and *disgust* as basic emotions, falls within this group.

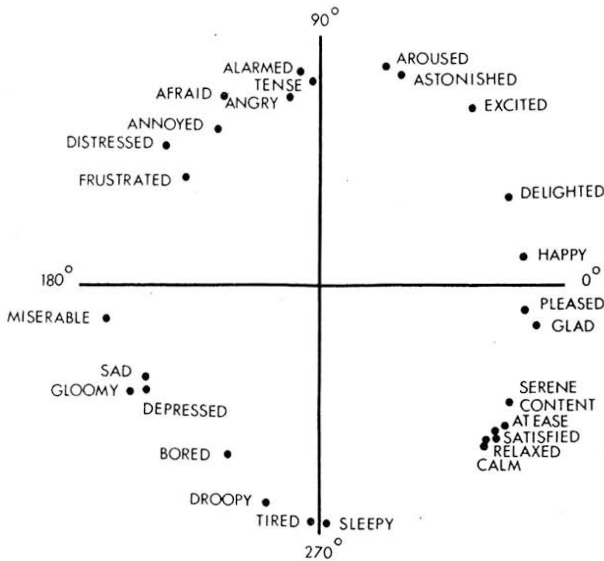


Fig. 4.34 Circumplex model of affect. Adopted from Russel (1980).

The second group describes emotion as the result of a number of continuous dimensions. Russel's circumplex model of affect (Russel 1980, shown in fig. 4.34) discerns two dimensions in emotion, termed *valence* and *arousal*. The valence axis corresponds to pleasant versus unpleasant emotions. To illustrate, happiness would have a high value for valence, whereas sadness would have a low value. The second dimension, arousal, correlates with the level of activation of the emotion. For example, surprise would score high on this axis, whereas tiredness would score low. Other models, such as the PAD model (Mehrabian, 1995) or the arousal-valence-stance model used in the robot Kismet (Breazeal, 2003b) expand upon this model by adding a third dimension. In Mehrabian's PAD model, this third dimension – the dominance axis – indicates a measure of control (or lack thereof)



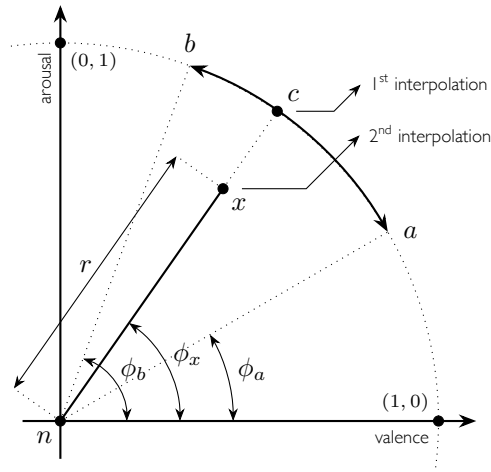
over others or situations. The stance dimension is used by Breazeal to indicate willingness to interact, from withdrawn (low stance) to approaching (high stance).

Finally, the third group of models uses a hybrid approach, employing both discrete categories and continuous dimensions to describe emotion. For example, Plutchik's wheel of emotions (1980) divides emotion into eight, primary, bipolar emotions, analogous to a color wheel.

The algorithm that the Opsoro system uses to generate facial expressions is based on the circumplex model of affect (Russel, 1980). Being a two-dimensional model, it is easy to manipulate and interact with in software. Secondly, as an advantage of using continuous dimensions, an emotion – and thus a facial expression – is defined for every 2D coordinate within the unit circle, facilitating smooth transitions between different facial expressions. Finally, this approach builds upon experience from within our research group, such as the work done on the affective system of Probo (Saldien, 2009; Saldien et al., 2010).

#### 4.3.1.1 BASIC ALGORITHM

Our algorithm is based on Russel's circumplex model (1980). It is advanced enough to accurately portray a wide range of emotions, though simple enough to be easily implemented in an software. In addition, it offers a continuous spectrum of emotion, enabling smooth transitions from one emotional state to another. In essence, our algorithm takes a number of pre-defined *key frame* facial expressions and mixes them together using a two-step linear interpolation process. This way, a facial expression can be generated for every point contained within the unit circle.



**Fig. 4.35** Two-step interpolation of the circumplex model

The algorithm takes two input parameters, numerical values for valence and arousal ranging from  $-1.0$  to  $+1.0$ , and outputs a servo pulse width value for one DOF. This process

is repeated for each DOF in the system, with different mapping data for every actuator, leading to a fully posed face. In practice, the process involves a number of steps. To start, the input parameters are translated from cartesian coordinates  $x$  equal to valence and  $y$  equal to arousal, to polar coordinates  $r = \sqrt{x^2 + y^2}$ ,  $\phi = \text{atan2}(y, x)$ . These polar coordinates can be intuitively interpreted as follows:  $\phi$  is a measure for the type of emotion, and  $r$  represents the emotion's intensity, from neutral (0.0) to maximum intensity (1.0).

In the next phase, the algorithm uses a two-step interpolation to determine the DOF position value of each DOF. The DOF position value ranges from  $-1.0$  (minimum position of the actuator) to  $+1.0$  (maximum position of the actuator). To do this, each DOF has a DOF map associated with it. This map contains a neutral DOF position (e.g.  $p_n = -0.2$ ), plus a number of DOF values at the edge of the unit circle (e.g.  $p_{30^\circ} = 0.6$ ,  $p_{45^\circ} = 0.8$ , ...).

For a given value of  $\phi_x$ , the algorithm looks up the two closest  $\phi$  values in the DOF map – with  $\phi_a < \phi_x$ , and  $\phi_b > \phi_x$  – to calculate  $p_c$ . In the first interpolation step, a value  $p_c$  is calculated using equation 4.4.

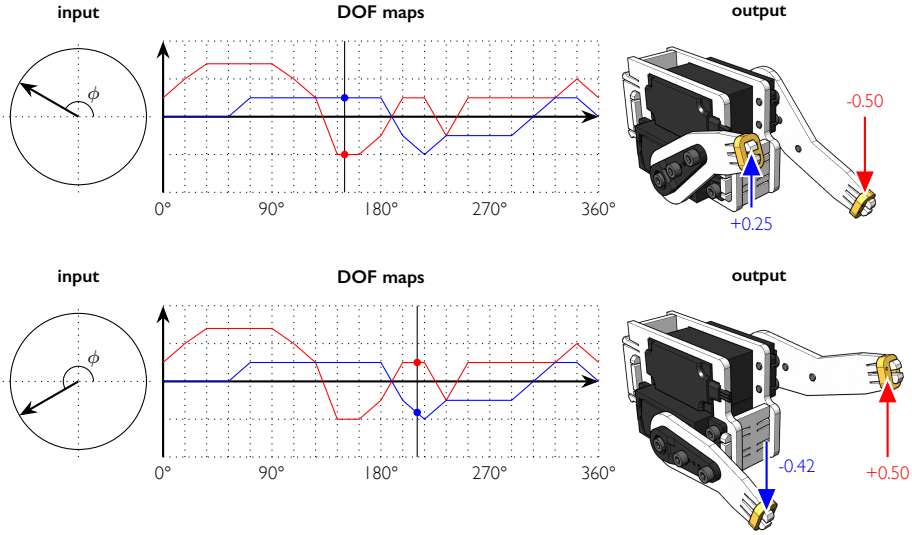
$$p_c = p_a + (p_b - p_a) \cdot \frac{\phi_x - \phi_a}{\phi_b - \phi_a} \quad (4.4)$$

Once a value has been calculated for  $p_c$ ,  $r$  is used as a scaling factor in order to perform a second interpolation step between  $p_n$  and  $p_c$ . As  $r$  is already a value ranging from 0.0 to 1.0, it can be used directly. Equation 4.5 shows how the final value of  $p_x$  is determined.

$$p_x = p_n + (p_c - p_n) \cdot r \quad (4.5)$$

It should be noted that this two-step interpolation process is performed for every DOF of the robot, resulting in as many DOF positions. Figure 4.36 shows the main steps of the facial expression algorithm applied to a simplified example. The example shows the two DOFs of an eyebrow module. The left-hand side shows the algorithm's input, namely a vector representing the desired emotion. In the middle of the figure, a graphical representation of the DOF maps is given. Each DOF of the system has one specific mapping function associated with it. In this case, the red line is associated with the inner eyebrow servo and the blue line is associated with the outer eyebrow servo. The input emotion vector is converted to polar coordinates, and the  $\phi$  value is used to look up the DOF positions in the DOF maps. These values are then used to drive the position each servo.

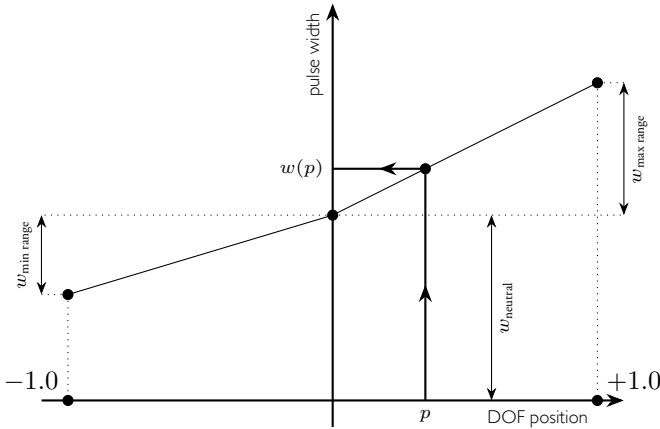
The final phase of the algorithm takes the DOF position value of each DOF and maps it to a pulse width value, which is then sent to the PWM controller. Because the neutral position for every servo is slightly different, and because not every servo or mechanism supports the same range of motion, three configuration values are used per DOF, indicating pulse widths corresponding to the neutral position, the maximum range of motion, and the minimum range of motion. These three values are used in the final translation



**Fig. 4.36** Example of the algorithm applied to a 2-DOF eyebrow module

step from DOF position  $p$  to pulse width  $w$ . Equation 4.6 shows how this is done. Figure 4.37 shows a graphical representation of the equation.

$$w(p) = \begin{cases} w_{\text{neutral}} + p \cdot w_{\text{max range}} & \text{if } p \geq 0.0 \\ w_{\text{neutral}} - p \cdot w_{\text{min range}} & \text{if } p < 0.0 \end{cases} \quad (4.6)$$



**Fig. 4.37** Mapping DOF positions to pulse widths

Note that the values for  $w_{\text{max range}}$  and  $w_{\text{min range}}$  do not necessarily have to be positive. By

defining  $w_{\max \text{ range}} < 0$  and  $w_{\min \text{ range}} > 0$ , one can make it so that the minimum DOF position corresponds to a longer pulse width than the maximum DOF position. This is useful if one wants to reverse the direction of rotation of a servo in software.

#### 4.3.1.2 C++ IMPLEMENTATION

The first iteration of the algorithm – intended to be run on a microcontroller, as explained in section 4.2.1 – was implemented in C++. Due to the constraints of the embedded environment, the DOF maps are stored at fixed steps of  $18^\circ$ , resulting in 20 circumference DOF positions plus one neutral position. All DOF maps are stored in a fixed-size 2D array. The algorithm was run on an early version of Ono, which contains 13 servos. Consequently, the DOF maps are stored in a  $21 \times 13$  array. DOF positions are also scaled from  $[-1.0, +1.0]$  to  $[-100, +100]$ . A sample of this 2D array is given in snippet 4.1.

**Snippet 4.1** 2D array containing DOF maps

```

1  const int Faces[21][13] = {
2      //          L_BROW_OUTER  L_BROW_INNER  R_BROW_OUTER  ...
3      /* 0: neutral */ {0,          0,          0,          /* ... */},
4      /* 1: valence */ {0,          25,          0,          /* ... */},
5      /* 2: happy   */ {0,          50,          0,          /* ... */},
6      /* ...         */ { ...         ...         ...         /* ... */},
7      /* 18: relaxed */ {0,          25,          0,          /* ... */},
8      /* 19: serene  */ {25,         25,          25,         /* ... */},
9      /* 20: contented */ {25,        50,          25,         /* ... */}
10 };

```

To interface the DOF maps with their respective servos, a helper class – appropriately named DOF – was created. Objects of this class store the pin number and minimum/maximum/neutral pulse width positions. The DOF class has a `DOF::SendPos()` method, which can be called in order to convert DOF positions to pulse width values, and to send these values to the correct servo. During the program initialization, an array of DOF objects is created and initialized with servo data, as shown in snippet 4.2. Note that the indices of the DOF object array correspond to the indices of the array containing the DOF maps.

**Snippet 4.2** DOF object array

```

1  DOF DOFs[13] = {
2      //  PIN  MIN  MID  MAX
3      DOF(0,   1800, 1500, 1200), // L_BROW_OUTER
4      DOF(1,   1300, 1600, 1900), // L_BROW_INNER
5      DOF(3,   1200, 1500, 1800), // R_BROW_OUTER
6      DOF(4,   1800, 1500, 1200), // R_BROW_INNER
7      DOF(6,   1800, 1500, 1200), // L_EYE_LID
8      DOF(7,   1150, 1500, 1750), // L_EYE_HOR
9      DOF(8,   1900, 1600, 1300), // L_EYE_VER
10     DOF(9,   1200, 1500, 1800), // R_EYE_LID
11     DOF(10,  1250, 1500, 1850), // R_EYE_HOR
12     DOF(11,  1300, 1600, 1900), // R_EYE_VER
13     DOF(12,  2100, 1500,  900), // MOUTH_L
14     DOF(13,   900, 1500, 2100), // MOUTH_R
15     DOF(14,  1800, 1500, 1100) // MOUTH_MID
16 };

```

At regular intervals, the program will determine the index associated with the current value of  $\phi$ , so that equation 4.7 is satisfied.

$$(idx - 1/2) \cdot \frac{360^\circ}{20} < \phi < (idx + 1/2) \cdot \frac{360^\circ}{20} \quad (4.7)$$

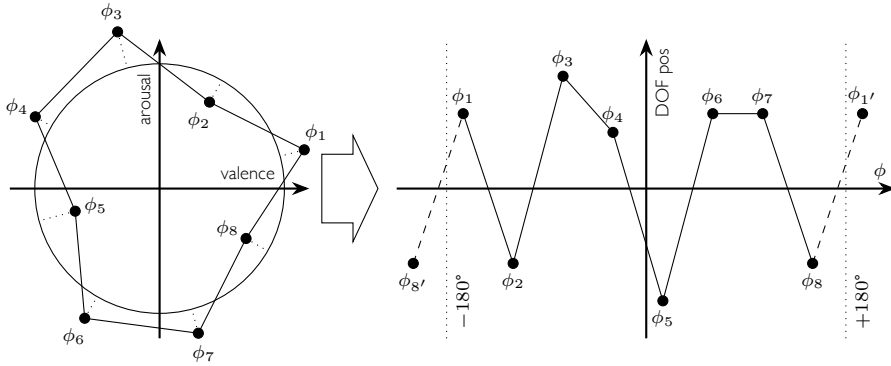
The program will then perform the two-step interpolation algorithm – as described earlier – by first interpolating between `Faces[idx+1][ ]` and `Faces[idx+2][ ]`, using  $(\phi / \frac{360^\circ}{20}) \bmod 1.0$  as interpolation factor. The second step interpolates between the previous result and `Faces[0][ ]` (the neutral facial expression) using  $r$  as the interpolation factor in order to calculate the final DOF position for that DOF. Finally, `DOF::SendPos()` is called in order to actually command the servo to move. Naturally, above steps are executed for each of the 13 DOFs.

#### 4.3.1.3 PYTHON IMPLEMENTATION

The second iteration of the facial expression algorithm – designed to run on the Raspberry Pi – was implemented in Python, a high-level, dynamic programming language. This language has a number of advantages from a programming perspective – such as dynamic typing and automatic memory management – but requires more processing power than low-level languages such as C++. Hence why a microcomputer running a full OS is needed, and a microcontroller is no longer adequate.

Switching to a more powerful platform afforded us the opportunity to make a number of improvements to the implementation. One of the notable changes is the way DOF maps are defined and stored in memory. Previously, DOF maps were restricted to 21 fixed points – one neutral value plus 20 values evenly spaced on the unit circle – due to the use of fixed-size arrays. Dynamic memory allocation and thus variably sized arrays is possible in C++. However, this is inadvisable because of the severe memory restrictions of embedded platforms, potentially resulting in unstable behavior of which the cause can be very hard to track down. In Python however, the basic array data type, a `List`, resizes automatically to accommodate more items. Consequently, the maximum number of points defining a DOF map is no longer fixed, allowing more or less data points to be added as needed. The size of different DOF maps is also no longer required to be equal to one another. A simple DOF may require a DOF map with only few data points, whereas a more complex DOF in the same robot can have many more points.

The Python implementation of the algorithm relies upon the *NumPy* and *SciPy* modules, which provide efficient numerical algorithms for tasks such as interpolation. Among other things, SciPy offers the `interpolate.interp1d()` function, enabling efficient linear interpolation using two sorted arrays (i.e. one for  $x$  data, another for  $y$  data). This function, though very useful and efficient, requires the DOF map data to be pre-processed, though this happens during program initialization and does not impact runtime efficiency. The first step is to constrain all  $x$  coordinates to  $[-180^\circ, +180^\circ]$ , this is done by taking the modulo of each phi value, and subtracting  $360^\circ$  from the result if the result is larger than



**Fig. 4.38** Interpolation using the NumPy module

$180^\circ$ . To go from periodic data (fig. 4.38, left) to a non-periodic interpolation function (fig. 4.38, right), two *ghost* points need to be added just outside of the  $[-180^\circ, +180^\circ]$ . The first ghost point, labeled  $\phi_{8'}$  in fig. 4.38, is a duplicate of the point with the largest  $\phi$  value. It is positioned at  $\phi_a - 360^\circ$ . The second ghost point ( $\phi_{1'}$  in fig. 4.38), conversely, is a duplicate of the lowest  $\phi$  value point, positioned at  $\phi_b + 360^\circ$ . The net result is that the interpolation function effectively wraps around over the  $[-180^\circ, +180^\circ]$  range. Then, all  $\phi$  coordinates are converted to radians, so as to facilitate manipulation of emotion using complex numbers. Finally, all points are sorted using  $\phi$  as a key, and the results are fed into `interpolate.interp1d()`.

A second point of change is the addition of *overlay* functions. These functions are optional callback functions that can be attached to one or more DOFs. They take a calculated DOF position and the associated DOF object as arguments, and can choose to return a modified DOF position. Excerpt 4.3 shows an example overlay function that periodically modifies the eyelid position in order to enable blinking. This approach preserves the circumplex model algorithm as a basis for the animation of facial expressions, but enables the option to move beyond that, should a certain application require it.

#### Snippet 4.3 Example overlay function

```

1 def blink_overlay(dof_pos, dof):
2     # Overwrite eyelid position to enable blinking
3     if blink_anim() > 0 then
4         return -0.8
5     else:
6         return dof_pos
7
8 Expression.dofs['L_E_LID'].add_overlay(blink_overlay)
9 Expression.dofs['R_E_LID'].add_overlay(blink_overlay)

```

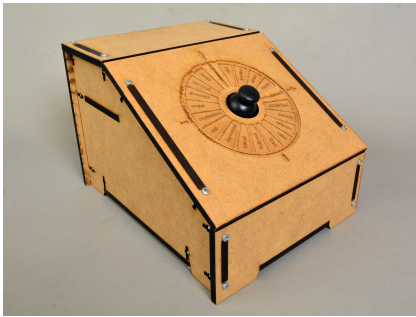
Finally, a minor point of change, robot-specific data (e.g. servo pins, range of motion, DOF maps) is stored in separate configuration files instead of being included during program compilation. YAML is used for these configuration files. This was chosen as a file format because it is light-weight, plain-text, and human-readable. As of right now, robot

configuration is split over three separate files. The first, “*pinmap.yaml*”, contains an associative array linking DOF names to pin numbers on the servo controller. The second configuration file, “*limits.yaml*”, contains the pulse width data describing the range of motion of each DOF. Finally, “*functions.yaml*”, contains the DOF maps of each DOF. Optionally, this file can also contain arbitrary additional data, which is loaded into the DOF class objects. This data is also accessible from within overlay functions, which could be used to specify overlay parameters.

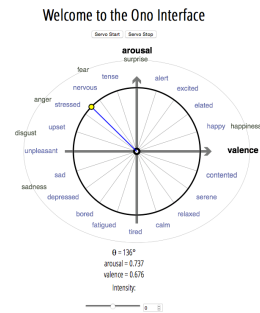
### 4.3.2 USER INTERFACE PRECURSORS

The first versions of the platform’s user interface were designed with the sole goal of allowing the hardware side of the platform to be tested, as mentioned earlier in section 4.2.1. Consequently, These interfaces can be seen as rudimentary at best. Nevertheless, for sake of comprehensiveness, a brief description is included in this section.

Two versions of this rudimentary interface were made, corresponding to the first generation of electronics (section 4.2.1) and the first iteration of the second generation (section 4.2.2.1) respectively. They were based around the concept of direct manipulation of the emotion vector (valence-arousal coordinates) by the user, allowing them to select an emotion, which the robot then conveys through facial expressions. This user interface is very primitive and offers little intelligence or interactivity, though it suffices for testing purposes.



**Fig. 4.39** Hardware interface



**Fig. 4.40** Web interface

The first version, shown in fig. 4.39, features a physical interface, built around a joystick. The joystick is positioned in the center of a diagram representing Russell’s circumplex model (Russel, 1980). The positions of Ekman’s six basic emotions (Ekman, 1992) are also indicated on the diagram.

The second version, shown in fig. 4.40, is a software implementation of the same interface. This version served as a proof-of-concept for the software architecture of the SBC-based electronics. The interface is implemented as a web page, and is served directly from the Raspberry Pi over a local WiFi network. The client side (i.e. the web page running on a user’s computer) communicates with the server side (i.e. the Python-based web server

running on the Raspberry Pi) using Asynchronous Javascript and XML (AJAX) calls. This method allows for communication between the two sides without reloading the web page every time data is transmitted. The architecture of the SBC-based software is described in more detail in section 4.3.3.2 and onwards.

4.3.3 APP-BASED WEB INTERFACE

The following sections describe the final iteration of the software platform. Influenced by prior experiences, we have made an effort to create an environment that is robust, easy to use (especially by non-experts), suitable for many kinds of social robots, and easily extensible to allow for new functionality ( $G_1$ ,  $G_{10}$ ).

4.3.3.1 DESIGN PHILOSOPHY

The initial Ono prototypes were very simple to operate, which proved beneficial for demonstrations and exhibitions, though it did not offer adequate functionality for practical use in experiments. For this reason, an extensible software framework was created to accommodate the various real-world use cases. Previous experience has made us keenly aware of the importance of the usability of the robot’s software. Very often, the robots will need to be operated by users without a technical background, such as a therapist or an educator. Users need to be able to turn on and use the robot, and the software should “just work”. With Probo, technical support staff was often required during use, a situation we wanted to avoid.

For this reason, we created a custom web-based interface running locally on a Raspberry Pi inside the robot. The operating system is configured so that a WiFi access point is created when the robot is turned on. Users can connect to the web interface through this network to control the robot. In addition to laptops, the robot can also be controlled using tablets and smart phones. Users do not need to install software on their system, avoiding potential practical problems.

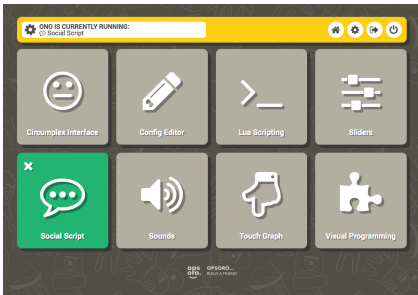


Fig. 4.41 Main page of the Opsoro interface, showing all apps

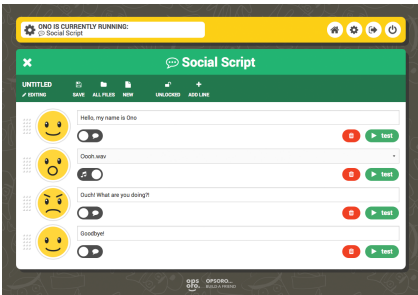


Fig. 4.42 Example showing the interface of the “Social Script” app



In recent years, modern web standards have advanced to a state where they support the creation of rich user interfaces. Contemporary web-based applications that rival the functionality of offline programs, such as Google Docs, is proof of this. Our web-based interface borrows the “app” metaphor of tablet and smart phone interfaces. Opening the main page of the interface, users are presented with a grid of apps (see figure 4.41), where each app is related to performing one specific task or scenario (see figure 4.42).

New apps can be created through a Python-based API. Our reasoning for this is twofold. Firstly, we cannot hope to anticipate all potential uses of the platform. Many potential applications for social robots are situated firmly in niche areas. These niche areas are also characterized by large amounts of tacit knowledge. Knowledge that is difficult to transfer from robot user to robot designer. Hippel (2001) coined the term “sticky information” to describe this problem, and proposes the design of toolkits as one solution. In this context, the software API is part of our toolkit for social robots.

The second reason is related to the first. Potential applications are very diverse, and ranges from therapy, to entertainment, to education and more. Simply put, we do not have the resources to fully support all these target groups. We chose the app-based software architecture as a workaround for this problem, allowing users to easily modify and extend the platform as they see fit ( $G_1$ ). The app-based system allows end-users to enhance the functionality of their robot simply by downloading a file. Because facial expressions are generated from valence and arousal parameters instead of being hard-coded, apps are independent from the robot’s embodiment. Consequently, the same software, the same apps and the same content can be used for many different kinds of robots, facilitating sharing and reuse ( $G_{10}$ ).

Figure 4.43 shows an overview of the platform software architecture. The following sections describe each of the elements in detail, starting from the OS level and building up from there.

#### 4.3.3.2 OS CONFIGURATION

On the most basic level, the software architecture relies upon the Linux operating system. The OS handles functionality such as task scheduling, file management, networking and peripheral control. Raspbian is used, which is the standard Linux distribution for the Raspberry Pi. Two changes have been made from the default OS configuration.

The network configuration has been changed so that the WiFi dongle starts in access point mode, creating a local network over which the user can operate the robot. Additionally, an mDNS service is configured so that the Raspberry Pi can be accessed through the “*ono.local*” URL, instead of requiring the end user to enter the device’s IP address manually. As a result of this configuration, the Raspberry Pi can function headlessly, and can be controlled without keyboard and screen attached.

The second set of modifications relate to the kernel module configuration. Modules are loaded that enable all the functionality of the Raspberry Pi’s 40-pin expansion header,

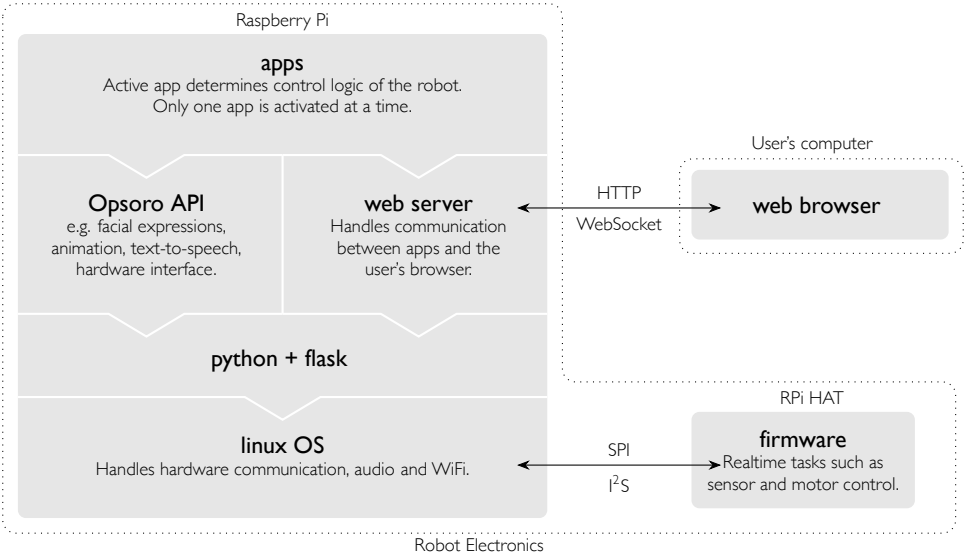


Fig. 4.43 High-level overview of the Opsoro software architecture

allowing access to the I<sup>2</sup>C, I<sup>2</sup>S, and SPI buses, as well as GPIO pins. The I<sup>2</sup>S bus is used to communicate with the audio circuitry on the daughter board. The SPI, I<sup>2</sup>C, and GPIO modules are used to communicate with the daughter board's microcontroller. The SPI bus is the main channel through which board functionality is controlled, including servo and sensor control.

### 4.3.3.3 WEB SERVER

The bulk of the software is written in Python, a high-level interpreted programming language. The main script is configured as a Linux daemon, a service that runs automatically at startup. This makes the interface accessible whenever the robot is turned on. The web server responsible for generating and transmitting the interface's Hypertext Markup Language (HTML) pages to the user's device is also part of the Python application. The Flask web framework<sup>6</sup> is used, along with the Tornado networking library<sup>7</sup> to handle the underlying communication. Tornado is used instead of Flask's built-in server in order to support WebSockets in addition to HTTP. WebSockets are used to provide a low-latency communication channel between client and server. This is needed for cases where the robot should immediately respond to interface input, or where sensor data needs to be continuously streamed from the robot to the user's device. HTTP is used mainly for the transmission of bulk data, such as the interface's HTML pages and assets.

<sup>6</sup>Flask (A Python Microframework) – <http://flask.pocoo.org>

<sup>7</sup>Tornado Web Server – <http://www.tornadoweb.org>

#### 4.3.3.4 USER AUTHENTICATION

To simplify the logic of the application, as well as to reduce the strain on the limited resources of the Raspberry Pi, the Opsoro interface is limited to one active user at a time. Having multiple users connected to the interface at the same time could result in situations where they send conflicting commands to the robot, leading to unexpected behavior. The programming overhead to deal with issues such as the arbitration of conflicting commands and interface state mirroring between connected clients was deemed to great, especially considering that we have never felt the need to control the robot from multiple devices in previous experiments.

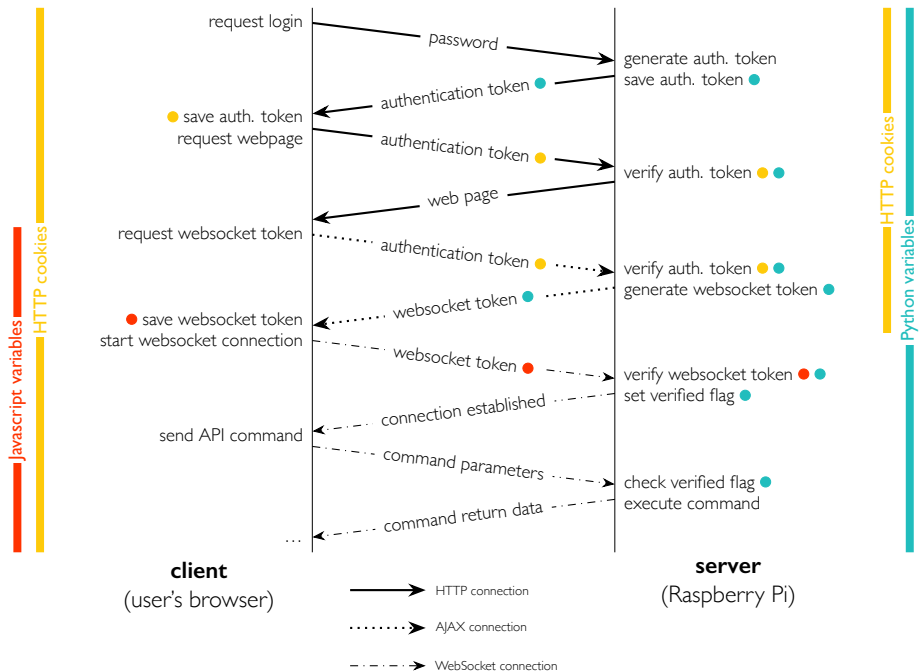


Fig. 4.44 Authentication scheme

Consequently, an authentication scheme (fig. 4.44) was devised to enforce single-user use. When a user first connects to the interface web server, they are redirected to a login page. Upon successful login, the server generates a string of 24 random bytes that serves as authentication token. This token is kept in memory on the web server, but is also stored on the client side as a session cookie. Every time a HTTP request is issued by the client, the server compares the client's session cookie to the token in memory. If they do not match, the user's session is destroyed and the request is redirected to the login page. In practice, this means that the most recently logged in user always has priority, as the value of the token on the server-side is overwritten each time a login occurs, invalidating any previously connected clients.

The algorithm to authenticate WebSocket connections builds on top of this. The session cookie token approach cannot be used directly because session cookies are not available from within the WebSocket context. Consequently, a work-around method was devised. First, the client requests a WebSocket authentication token from the server using an asynchronous HTTP request. The server uses the session cookie token to validate the authenticity of the request, and generates and returns a new token for WebSocket authentication. In the second phase, the client opens a WebSocket connection to the server and transmits the token it received in the HTTP request, at which point the WebSocket connection is authenticated.

It should be noted that the authentication methods described in the above paragraphs are not waterproof by any stretch of the imagination. They serve primarily to deal with situations where multiple users are simultaneously connected by accident. A sufficiently motivated malicious user could probably find exploits in these systems, though in our context this scenario seems unlikely, and the impact of such a hack would remain limited to the robot.

#### 4.3.3.5 APP MANAGER

For similar reasons as to why only one user can control the robot, we chose to make it so only one app can be activated at a time. Having multiple apps active simultaneously would require all apps be programmed in such a way that they negotiate for control of the hardware among themselves. Seeing that we want the software to be extensible using third-party apps, we cannot predict all inter-app interactions. Consequently, the software imposes a constraint that only one app can be activated at a time.

The app manager of the software performs a number of different tasks. Firstly, it handles the discovery of installed apps. Apps are self-contained within their own folder, and containing one or more Python scripts, plus all client-side files required for the app's user interface, such as HTML templates, CSS files, images, and Javascript files. These folders are automatically loaded as modules by the app manager and relevant information, such as the app's name and icon, is extracted automatically for use in the interface.

Secondly, the app manager lets apps define web pages within their own `/apps/<app name>/` URL path. The app manager automatically injects the glue logic required to check the user's authentication and to redirect the user to the main page if the web page belongs to an inactive app.

The final function of the app manager is to handle activation and deactivation of each app, ensuring that the app's `start()` and `stop()` methods are called at appropriate times. App authors can choose to overwrite these methods to handle whatever activation/deactivation logic they require, such as turning servos on/off or starting/stopping a separate program thread.

Figures 4.45, 4.46, and 4.47 illustrate the functionality of the app manager in various situations. Figure 4.45 shows the initialization of the system. During this phase, the app

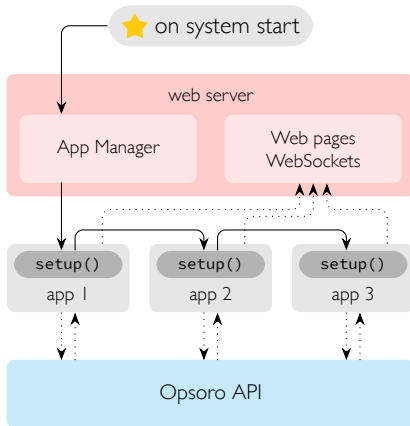


Fig. 4.45 App manager – setup

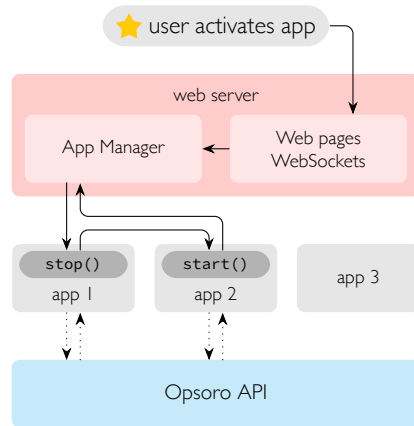


Fig. 4.46 App manager – activation

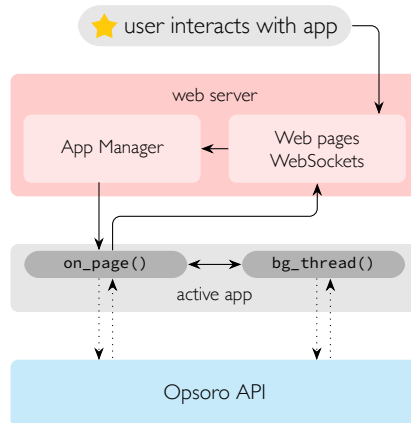


Fig. 4.47 App manager – interaction

manager auto-detects the installed apps and sequentially runs their `setup()` functions, allowing the apps to register webpages, register WebSocket handlers, and perform any necessary initialization steps. Figure 4.46 shows how app switching is handled. First, the currently running app's `stop()` method is called, allowing it to release control to the hardware. Then, the new app's `start()` method is called, signaling the start of that app. Finally, figure 4.47 shows the general workflow of an active app. Here, the app manager is responsible for redirecting web events to the appropriate handler inside the active app. In addition to generating a web page, this handler can also interact with the API, for instance to change the facial expression of the robot. The figure also shows an optional background thread. This thread can be used for behaviors that are not the direct result of operator input, for example eye blinking or responding to touch sensor events.

### 4.3.3.6 HARDWARE AND EXPRESSION

The software also provides a number of modules that are intended to be used by apps. The most important ones are the Hardware module and the Expression module. These modules provide simplified access to raw hardware control and the facial expression algorithm respectively.

The Hardware module provides access to the servos (e.g. turn on, turn off, set position), to the capacitive touch sensors (e.g. initialize channels, get raw/filtered data, detect touch), and to various utilities (e.g. reset microcontroller, get board version, turn LED on). This all happens over the SPI bus, using the protocol described in section 4.2.2.2, which is also implemented by the module.

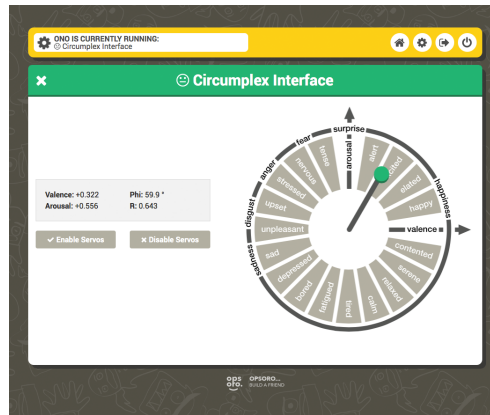
The Expression module encapsulates the facial expression algorithm described in section 4.3.1. It handles the loading of robot-specific DOF and DOF-map data from configuration files, and uses the Hardware module to output facial expressions based upon input from apps. Apps can control facial expressions using either valence and arousal coordinates or  $r$  and  $\phi$  coordinates. Apps can also choose to overwrite certain DOFs using overlay functions.

### 4.3.3.7 UTILITY MODULES

In addition to the Hardware and Expression modules, the software also provides a number of utility modules for apps to use. The Sound module provides simplified access to audio file playback, speech synthesis, and volume control. The Console Message module provides various methods for outputting debug messages. The Animate module provides helper classes to simplify programming animations. The classes provide time-based linear interpolation of scalar values based on keyframes, though it is left up to the app author to link the animation to an output, such as a DOF position. Finally, the Stoppable Thread module provides a way to let a function run indefinitely in a thread independent from the web server, though the thread can be stopped at request from the main application. This is useful for making certain actions happen continuously and independently from user input, for instance a periodic eye blink animation.

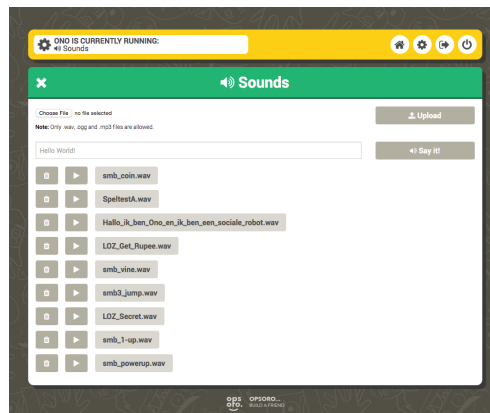
### 4.3.3.8 OVERVIEW OF APPS

Over the course of software development, a number of first-party apps were created. This includes both apps that are more testing-oriented as well as apps that are designed for user interaction. This section aims to give an overview of all currently existing apps and their functionality.



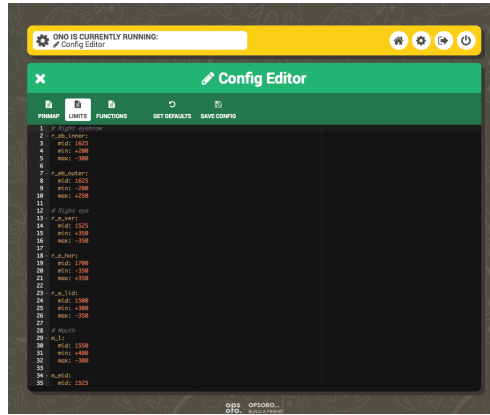
**Fig. 4.48** Circumplex Interface App

The Circumplex Interface app (fig. 4.48) allows the user to manually control the facial expression of the robot using an interactive circumplex model figure, as described in section 4.3.2. Valence, arousal,  $r$ , and  $\phi$  values are shown so that they can be copied for use in other apps or scripts. Basic animation is used so that facial expressions transition smoothly from one emotion to the next.



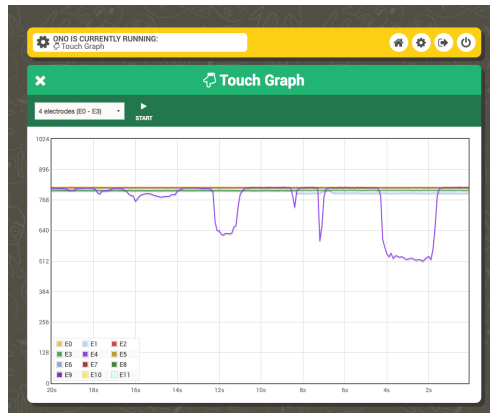
**Fig. 4.49** Sounds App

The Sounds app (fig. 4.49) can be used to manage and test the audio-related features of the robot. Text can be entered directly to test the text-to-speech functionality of the software. The user can also upload and play back audio files from the interface. Audio files uploaded in this app are made available to other apps, such as the Visual Programming app and the Social Script app.



**Fig. 4.50** Config Editor App

The Config Editor app (fig. 4.50) allows the user to edit the system configuration YAML files directly from their browser. These files include the pin mapping, which correlates DOF names to the physical I/O pins for servos, the servo limits, which define the range of motion for each DOF, and the DOF mapping, which defines how valence and arousal parameters are transformed into DOF positions. This app was used for testing. Seeing that the text-based configuration language is not particularly user-friendly, a GUI-based version will need to be developed in the future.



**Fig. 4.51** Touch Graph App

The Touch Graph app (fig. 4.51) is used to test and debug various electrode configurations of the capacitive touch sensor IC. The app continuously plots the sensor values of 1 to 12 electrodes, allowing users to iteratively test the robustness and sensitivity of home-made capacitive touch pads. A stoppable thread is used to read sensor data at a frequency of 10 Hz. Sensor data is streamed from the robot to the browser using a WebSocket.



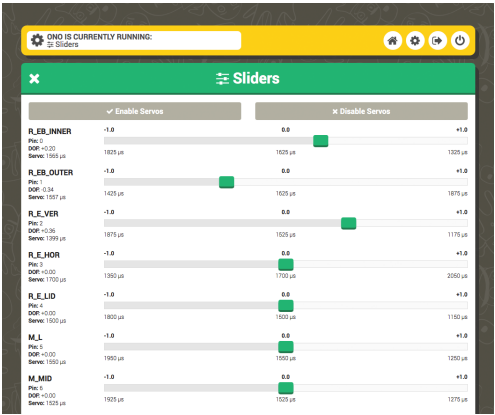


Fig. 4.52 Sliders App

The Sliders app (fig. 4.52) lets users manually control servos using a set of sliders. The sliders drive the DOF position of each servo, ranging from  $-1.0$  to  $+1.0$ , and are consequently restricted to stay within a safe range. This app is useful for debugging problems with servos, as well as for fine-tuning the neutral position for each servo. Initially, AJAX requests were used to transfer DOF position from the browser to the server, though this approach suffered from problems as messages could be received by the server in a different order. The issue was solved by changing the communication method to a WebSocket, which guarantees that messages will be received in the correct order.

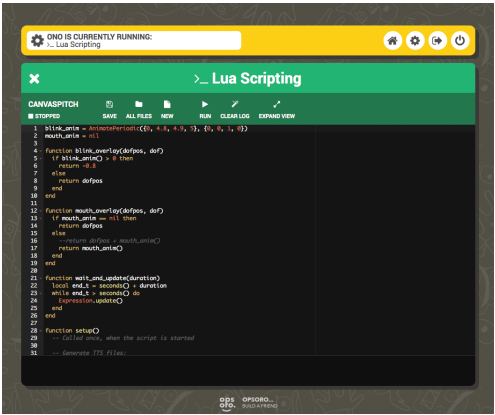


Fig. 4.53 Lua Scripting App

The Lua Scripting app (fig. 4.53) allows users to write and execute scripts from within their browser. The interface offers a text editor field with syntax highlighting, as well as buttons to save, load, and run files. The scripting environment itself uses the Lua programming language, and code is executed in a sandbox environment in order to protect

the main application from script bugs. The structure of scripts is simpler than that of a full app, requiring only a `setup()`, a `quit()`, and a `loop()` function, where users can insert initialization commands, deinitialization commands, and main program code respectively. Scripts have full access to the Hardware and Expression modules, as well as a separate UI module that lets script authors add buttons to the script interface, which can then be linked to the script code.

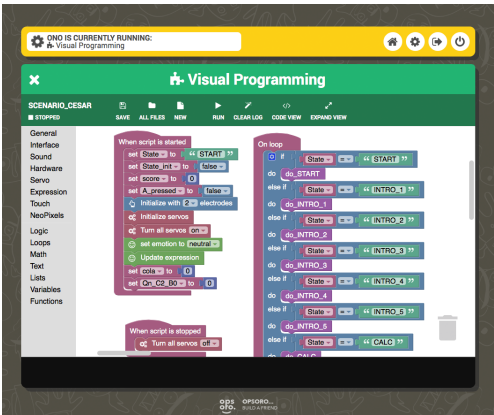


Fig. 4.54 Visual Programming App

The Visual Programming app (fig. 4.54) lets users author custom scripts by dragging and connecting visual blocks that represent various commands. The goal of this app was to offer a way for non-programmers to create complex custom scenarios. The visual programming language relies upon Blockly, a javascript library offering a visual programming environment derived from Scratch (Resnick et al., 2009). Blockly offers a standard set of programming blocks, such as an `if` block and a `print()` block. This set of blocks was extended to include Opsoro-specific blocks, such as a `set_emotion()` block and a `say_tts()` block. The environment translates block-based programs to Lua code, which is then transmitted from the browser to the robot and is executed using the script host of the Lua Scripting app.

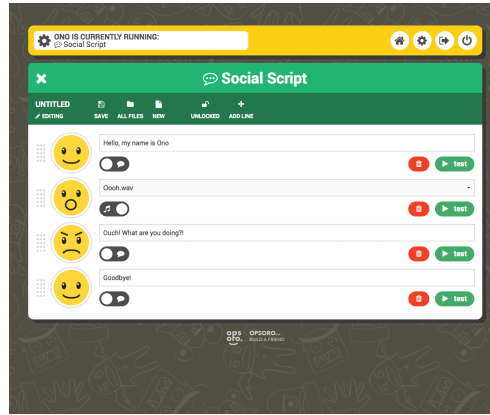


Fig. 4.55 Social Script App

The Social Script app (fig. 4.55) allows for an even simpler way of preparing dialogs, and is intended to be used in Wizard of Oz interaction scenarios. In authoring mode, a user can specify a number of voice lines. Each voice line consists of a facial expression coupled to either a sound file or a line of text intended for text to speech. In play mode, each of these lines becomes available as a clickable button that makes the robot change its facial expression and say the appropriate line of text.

#### 4.3.3.9 IMPLEMENTATION OF SCRIPTING ENVIRONMENTS IN APPS

Because user extensibility ( $G_1$ ,  $G_{10}$ ) is one of the core ideas behind the Opsoro platform, we have created a number of apps that focus on enabling novices to create complex behaviors using various scripting interfaces. The different incarnations of the various “scripting” apps taught us many valuable lessons, on both a technical level, as well as from the perspective of the various types of users. By supporting different degrees of scripting difficulty, skill development and flow is stimulated ( $G_6$ ).

The first attempt at creating a scripting environment app was the initial version of the Visual programming app, shown in figure 4.54. The app was created as part of the preparations of a one-day workshop where participants constructed new social robots by combining the platform’s modules with craft materials such as cardboard and foamcore. More details on the setup of the workshop can be found in section 3.5.1.

As is the case with the current version of the app, the initial version built upon the Blockly Javascript library<sup>8</sup> to implement the scripting environment, resulting in a very similar UI as the present version of the app. Architecture-wise, the differences are substantial however. The Blockly environment was configured to generate Javascript code from the

<sup>8</sup>Blockly – <https://developers.google.com/blockly/>

Blockly script. The generated Javascript code was executed in the browser, with the various platform-specific API calls linked to the server using AJAX requests.

There were a number of technical problems with this approach. To begin with, implementing a `sleep()` function was non-trivial, owing to the fact that the Javascript language has no simple way of suspending code execution. Instead, programs are expected to handle timing functionality using the `setTimeout()` function, allowing the programmer to specify a callback function to be run at a certain point in the future. We found the callback paradigm too complex for casual users, the `sleep()` model is much easier to comprehend and use in own programs. The issue was solved by using JS-Interpreter<sup>9</sup>, a Javascript interpreter written in Javascript. The interpreter lets users specify an arbitrary delay between program execution steps. This was exploited by creating a `sleep()` function that temporarily overwrites the step delay for one execution step.

The second problem was related to the AJAX API calls. At this point in time, WebSocket infrastructure had not yet been implemented. Consequently, all communication between the user's browser and the robot happened over AJAX calls. Outside of latency, the biggest issue with AJAX calls is they are not guaranteed to be received by the server in the same order as they were transmitted, as is implied by the *asynchronous* part of the technology's moniker. This poses problems for certain programs, for example when a `set_emotion()` call is followed by `update_emotion()`. Reversing the order of these two calls would result in the desired facial expression not being shown by the robot.

Finally, because the generated script runs in the browser and all API calls need to be transmitted over the network, an visible amount of latency is introduced. Facial expression animations became choppy due to the limited amount of "frames per second". Sensor inputs also proved less responsive, as the browser first had to request sensor data from the server, then process it, and then request the server to perform a certain action.

The second step forward was the creation of the Lua Scripting app, shown in figure 4.53. With respect to performance, the key difference with this app is that it runs scripts directly on the server, and any feedback is transmitted to the browser over a low-latency WebSocket connection. App users can design new behaviors for the robot by writing simple scripts in their browser. The browser-based editor offers syntax highlighting as an aid. Owing to the fact that a traditional programming language is used, this app is not so much suited for complete novices as it is for users who already know how to program. Still, the script structure and API are chosen so as to mirror the philosophy of "sketches", as understood by Processing (Reas and Fry, 2007) and Arduino (D. A. Mellis, Igoe, et al., 2007). Users do not program full-blown applications, but rather sketch out their ideas using a text-based language.

To execute the script, the browser transmits the script's source code to the server, where it is executed in a sandbox environment. The Lua programming language (Ierusalimsky et al., 2007) was chosen for this environment, bindings to the main python code are created

<sup>9</sup>JS-Interpreter: A sandboxed JavaScript interpreter in JavaScript. – <https://github.com/NeilFraser/JS-Interpreter>

using Lupa<sup>10</sup>. Lua was chosen for a number of reasons. It is lightweight, very fast, and designed to be embedded in larger applications. However, the biggest advantage is that Lua can be sandboxed easily, whereas the flexible nature of the Python language makes it very difficult to secure against malicious code.

In the case of the Lua Scripting app, a new `ScriptHost` class was created. The `ScriptHost` class exposes hardware and facial expression APIs functionality to the script while blocking access to potentially malicious functionality, such as file manipulation and shell access. The `ScriptHost` class is also responsible for running the script's `setup()`, `loop()`, and `quit()` functions, as well as stopping scripts by force, if necessary.

This Lua-based implementation of a scripting environment was later back-ported to the Visual Programming app. The design of the Blockly library is language-agnostic. Consequently, the code generation logic of the app was changed to output Lua code instead of Javascript code. Currently, the Visual Programming app runs scripts on the server, and communication between browser and server takes place over a WebSocket. This solved the technical problems of the previous version of the app.

User testing with therapists revealed a number of shortcomings of the Visual Programming app. During these tests, we helped therapists to implement simple interactive scenarios in the app. It became apparent that many of the blocks were too low-level, resulting in user confusion and excessively long scripts. For instance, to change the facial expression, one would first have to initialize the servo controller, then set the emotion, then update the emotion, and then turn the servos on. This makes sense from a programming perspective and allows for granular control of functionality, but is unintuitive for third-party users, as we discovered. Future versions of the app should encapsulate more commands in a single block so as to provide a higher-level interface for users to use. For example, initialization commands should simply be included in the `setup()` block without requiring user intervention, and the `set_emotion()` block should also immediately update the facial expression.

Analysis of the scripts revealed a frequently reoccurring pattern where a facial expression is linked together with a line of dialog or a sound file. From further interaction with the therapists we learned that a simple interface with buttons that change the facial expression and play some audio would be sufficient for most Wizard-of-Oz use cases. During a paper prototyping exercise, we arrived at an app concept where the therapists could prepare a list of dialog lines, consisting of an emotion together with either a line of text or a sound file. This way, relevant lines around a specific theme (e.g. bullying) could be prepared in advance, and then be played back by the therapist in real-time based upon the actions of the child. The concept was implemented in the Social Script app, shown in figure 4.55. The advantage of this app is that it is much simpler to understand than the other scripting apps, and that new scenarios can be prepared rapidly. However, the other scripting apps allow for much more complex behavior at the cost of a steeper learning curve.

<sup>10</sup>Lupa: Python wrapper around Lua and LuaJIT – <https://pypi.python.org/pypi/lupa>

## 4.4 CONCLUSION

This chapter has discussed the the technical implementation of the Opsoro platform, including the mechanical design of the modules and the embodiment, the electronics design of the driver board, and the software architecture. The design goals described in section 1.4 have impacted many of the technical design decisions. The components used in the Opsoro platform were specifically chosen to be inexpensive ( $G_9$ ) and readily available worldwide ( $G_1$ ), avoiding specialized high-performance components. Similarly, custom components can be built using production technologies that are appropriate for the modern DIY'er, such as lasercutting and 3D printing ( $G_1$ ). Efforts were made to make it easy to get the parts, but also to simplify the assembly process ( $G_2$ ). The platform builds upon the work of established open source software and hardware communities, such as those of Arduino and Flask ( $G_{10}$ ). Furthermore, the platform is designed to be extended by third parties; both in software through the app system and in hardware through the module system, further stimulating community involvement ( $G_{10}$ ).

The current prototype serves as a good basis to expand upon and allows us to quickly try out new ideas. Having working prototypes also greatly facilitates communication with different stakeholders, such as therapists, educators, and researchers, which helps to inform the next steps in development. Still, much development work remains in preparation of commercialization of the platform.

### 4.4.1 FUTURE ENHANCEMENTS

The experiment with the Opsoro Grid system (section 3.5.3) showed promising results. In the experiment, the mechanical interface of the Opsoro Grid proved more reliable than the snap connectors. This construction system will be further be further developed, taking building affordances and toolkit aesthetics into account. Current module designs will be updated to conform with this new mechanical interface. Furthermore, the system will be extended with new modules, such as LED eyes, touch sensor pads, and a neck module.

The Opsoro HAT board will be redesigned and optimized for manufacture. A small batch of 24 boards of the current HAT design has already been produced, and through this process we discovered a number of shortcomings. Currently, the design has too many components, driving up assembly cost and introducing more points of failure. A more integrated solution with less components can be achieved by moving to a better microcontroller, e.g. a 32-bit ARM device. Furthermore, during manufacture we discovered that the capacitive touch IC had reached end-of-life, meaning that this chip will have to be replaced. Finally, the power architecture of the system needs to be improved. For safety reasons, the new version should use a separate “power brick” adapter instead of an integrated metal-enclosure power supply.

By using the Opsoro API ourselves and by seeing others use it, we have discovered a number of improvement points. Over the past year, we have seen a number of recurring

program patterns. These patterns occur in many different apps and scripts, but currently need to be reimplemented on a per-app basis. Examples include file management (saving, loading, executing), autonomous behaviors such as blinking, and lip synchronization for speech. This functionality should be generalized and made available through libraries. Similarly, the visual programming environment should be redesigned so that related commands are grouped into higher-level constructs. For instance, the `say_tts()` command is almost always followed by a `sleep()` command. These commands should be integrated so that the text-to-speech functionality automatically waits until the audio has stopped playing.

Finally, one of the major remaining tasks is improving the documentation for building, modifying, and using Opsoro robots. In order to encourage an active community of users, better documentation should be offered. Current assembly documentation consists of step-by-step photos with short written instructions. We found that this is not always the most appropriate format, especially for complex steps. Assembly documentation should be enhanced with rich media such as video, exploded views, and embedded 3D model viewers. Software documentation should also be improved, and should include a tutorial series on designing a custom app.

#### 4.4.2 NEW DEVELOPMENTS

As of yet, the software stack is offline, running locally off of a Raspberry Pi. We want to extend the software to include an optional online platform. The intent of the online platform is to fulfill two major functions. First of all, it would enable users to easily share content, such as interaction scenarios, scripts and apps. Secondly, it would allow users to simulate the behavior of a robot using a virtual model, allowing them to experiment with the system without direct access to a robot. The online platform would thus allow them to prepare content in advance, and synchronize the data to the robot at a later stage. Work on developing this online platform is already underway. Because the software interface was originally designed using web technologies, this process has been relatively straightforward.

Furthermore, we wish to incorporate more advanced behaviors into the system, including more sophisticated animation, autonomous behaviors such as blinking and looking around, and basic artificial intelligence features. Some of these improvements will necessitate us to incorporate a camera, allowing us to use computer vision to things like face tracking, emotion mimicking and tag detection.

Currently, the software does not rely on, or expose an interface to ROS (Quigley, Conley, et al., 2009), a popular open source framework for robots control. The main reason for this choice is because the two projects have very different goals. Opsoro focuses on relatively simple, stationary robots with a limited number of DOFs and no advanced sensors (e.g. LIDAR). On the other hand, ROS was created for large-scale service robots, typically with multiple onboard computers, offloading intensive computations to a stationary server. ROS emphasizes autonomous behavior, whereas we focus on semi-autonomous scenarios

and Wizard-of-Oz scenarios. While there is some overlap between the two systems, different goals lead to very different points of attention. In our work, the majority of the code is related the user interface. Conversely, while ROS offers tools to create GUIs, these tools are primarily intended for debugging purposes, focusing on sensor graphs, console messages and environment data visualization.

Still, we do plan on introducing some degree of interoperability with ROS. The reason for this is twofold. Firstly, ROS is a well-known standard within robotics research. Many users are already familiar with it, and by providing a ROS interface, we facilitate the process of integrating an Opsoro robot as part of larger experiments. Secondly, integration would allow some of ROS's advanced functionality, for instance emotion recognition algorithms, to be used within an Opsoro robot. Lastly, we expect that integrating the ROS API will be fairly straightforward. ROS already includes bindings to the Python programming language. Furthermore, one of the third-party Ono users has already put together a preliminary ROS binding.

This chapter has summarized the technical details of the Opsoro platform, as well as future steps in technical development. The next chapter concludes this work with final reflections and a discussion of future prospects.



## Chapter 5

# CONCLUSION & FUTURE PERSPECTIVES

This dissertation has detailed the inception and development of Opsoro, a platform for DIY social robots. The platform enables amateurs and non-experts to design, build, and use custom social robots for face-to-face communication. With this work, we have made social robotics technology accessible to a wide audience of users by building upon contemporary DIY principles and practices, such as the open source hardware movement and the maker movement. Designing a platform is different than designing a single system, not only from a technical standpoint, but also because of the way they are used. For this reason, traditional engineering paradigms were eschewed in favor of an iterative, user-centered design process that emphasizes user experience aspects. Each iteration has led to a better understanding of different DIY approaches in the *design*, *build*, and *use* phases of the platform. The result is an inexpensive, open source, DIY-friendly platform that can be used to design, build and use custom emotionally expressive robotic characters, fulfilling all goals set forth in chapter one (section 1.4):

- *Open* ( $G_1$ ) – The platform is designed so it can be manufactured using maker-friendly production techniques (i.e. laser cutting and FDM 3D printing) and common, inexpensive components. This design choice makes it easy to copy, hack, and extend the platform.
- *Easy to build* ( $G_2$ ) – During the design of Opsoro, special attention was given to the assembly process; by using the characteristics of digital manufacturing to our advantage, we managed to create designs that are easy for novices to assemble and replicate.
- *Emotionally expressive* ( $G_3$ ) – The platform enables users to create custom social robots that support facial expressions, speech, and gaze as the main ways for emo-

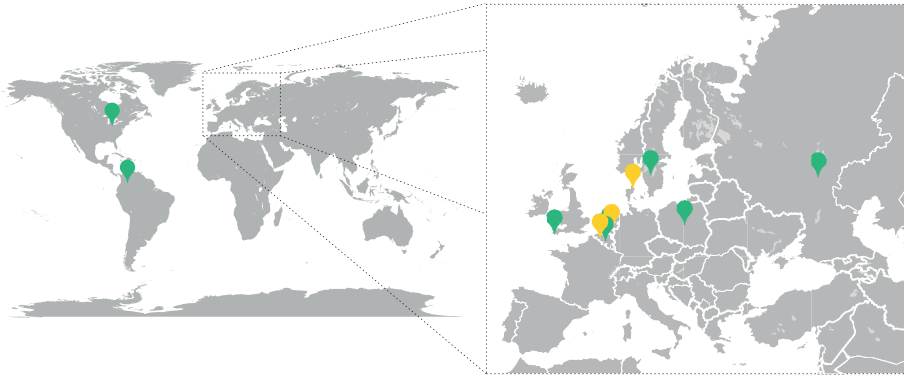
tional expression. The software is designed so that users can control the emotional expressions of their robot through simple and intuitive interfaces.

- *Facial features* ( $G_4$ ) – Through our iterative, user-centered approach, we have found a good balance cost and ease of use on the one hand, and advanced facial features on the other hand. To this end, we have designed flexible, low-cost modules that implement the different elements of a face.
- *Character* ( $G_5$ ) – In our platform, we focus on realizing iconic emotionally expressive characters instead of realistic, human-like robots. Drawing inspiration from cartoons and animation, we provide the tools to create artificial creatures rather than mechanical robots.
- *Creativity* ( $G_7$ ) – We emphasize the importance of open-endedness and creative play in our platform. We offer users an easy-to-use, extensible set of building blocks to experiment with, as well as a methodology to guide them through the creative process. Creativity is stimulated throughout the design of the toolkit, enabling a spectrum from minimal customization to fully custom-created designs.
- *Flow* ( $G_6$ ) – A state of flow is stimulated by offering an open-ended system that addresses users of all skill levels, offering multiple paths for hardware and software. The platform is designed so that it grows with the user. Opsoro is easy for novices to use, and the system gradually offers new possibilities as the user becomes more experienced.
- *Diverse knowledge domains* ( $G_8$ ) – The platform introduces STEAM-related skills and knowledge in a novel and unexpected format. In addition to the traditional robotics elements, the platform also integrates practices from crafts, design, and art.
- *Low cost* ( $G_9$ ) – By using low-cost digital manufacturing techniques such as laser cutting and FDM 3D printing, combined with inexpensive standardized components, we have created an affordable platform for social robotics. This way, social robotics technology is made accessible to a larger audience of users.
- *Community-oriented* ( $G_{10}$ ) – Our work builds upon existing technology from the open source community and our platform is also released under an open source license to further stimulate community involvement. Additionally, we are currently working on an online service as a companion to the Opsoro software. First steps have already been undertaken; an alpha version of the online community platform was tested in the last iteration of the Opsoro.

In our experiments, the platform was used by non-experts to design and build new social robot embodiments from scratch. The experiments encompassed many different types of users, representative of the different stakeholders in human-robot interaction, and human-computer interaction in a broader sense. The experiments also varied in time scale, ranging from only a couple of hours to over three months. The experiments cover the creation of new designs, the reproduction (copying) of existing designs, and the actual use of robots

in interaction scenarios. One of the most remarkable result of the experiments is that participants always successfully completed their task. At the start of experiments, participants frequently voiced their concerns that the challenge would be too difficult, that it was outside of their skill set. Nevertheless, they always succeeded, noting the process was much easier than it appeared. The robots built by participants during the experiments are very diverse, and participants often came up with novel and unexpected designs. We also note that participants quickly formed a bond with their creations, similar to the IKEA effect (Norton et al., 2012). They would always refer to the robot as *him* or *her* and not *it*, and they would frequently take pictures and selfies with the robot. We observed this phenomenon when they assembled a “standard” Ono, but the effect was even more pronounced when they were building their own designs. The Opsoro platform forms a bridge between robot designers and robot users by making a complex technology accessible for a wider audience. In summary, this project lays new foundations toward democratizing social robotics technology through a DIY, open source platform.

Technical development encompasses modular hardware building blocks, an embodiment design methodology, a specialized circuit board for the Raspberry Pi, and a web-based software interface to control the robots. All elements of the platform are published as open source software and hardware so that others can use and build upon our work. Throughout this work, we have shown how the DIY paradigm can impact social robotics research. We have shown how digital manufacturing technologies can drastically lower the barriers toward custom social robot designs, making it easy to quickly design a wide variety of different social robots. Within this dissertation, we focused on incorporating DIY approaches in social robotics research. However, these methods and techniques hold potential for many different domains, offering new ways to make complex technology accessible to a large audience of users.



**Fig. 5.1** Map of current users of the social robot Ono (yellow) and the Opsoro Starter Kit (green).<sup>1</sup>

On a personal level, I am proud to say that our work did not stay confined to our lab, but is

<sup>1</sup> *Blank map of Europe* by “maix”.  
Licensed under Creative Commons Attribution-Share Alike 2.5 Generic.  
*World map blank without borders* by “Crates”.  
Licensed under GNU Free Documentation License 1.2.

Ono robot users			Opsoro Starter Kit users		
2014	Russia	University of Nizhny Novgorod	2016	Denmark	Aalborg University (3 kits)
2015	Poland	Lodz Polytechnic University	2017	Belgium	Rhizo School
2015	Colombia	Colombian School of Engineering	2017	Belgium	VIVES University College (4 kits)
2015	USA	University of Michigan	2017	Netherlands	M.S. (private individual)
2015	UK	Plymouth University			
2015	Belgium	Vrije Universiteit Brussel			
2015	Sweden	University of Skövde			
2016	Belgium	University College of West-Flanders			
2016	Belgium	De Lift Education			
2016	Netherlands	Rotterdam University of Applied Sciences			

Table 5.1 Overview of current users



Fig. 5.2 OTO – a mobile extension for the Opsoro platform

being used by researchers and practitioners worldwide. Seeing the enthusiasm of children, therapists, and fellow researchers is one of the greatest rewards. As of yet, we have sold ten Onos to other universities and research institutes, spread over three continents (fig. 5.1, table 5.1). These robots are actively being used in a wide range of research, development and educational applications. For instance, Ono is used as a tool for speech therapy and behavior therapy experiments with children. Additionally, one of the Ono’s is currently being used as a development platform in computer science courses, which has resulted in the development of novel third-party apps for the Opsoro platform. Finally, one of the HRI studies performed using the Ono robot has already been published (Zubrycki

and Granosik, 2016), and one of the users is planning on building a second Ono from scratch, using the source files on GitHub. Currently, the Opsoro Starter Kits are used in secondary schools for STEM education purposes. The Opsoro platform is currently also used as a co-creation toolkit in a research project investigating novel social interfaces for people suffering from dementia. Within our own university, students have developed an extension for the Starter Kit allowing users to construct mobile, car-like social robots (fig. 5.2).

As of yet, HRI studies primarily investigate robot behaviors, ignoring the role of embodiment as part of the design space. We hope others will build upon our work, leading to better a understanding of the role of embodiment in social interaction with robots. Within this dissertation, the scope of the study was focused on the design of DIY robotic characters with actuated facial features. Others have focused on designing open source social robots that emphasize body motion (e.g. Poppy). An open question is how these two aspects can be united through a DIY approach within the same platform. Undoubtedly, combining both approaches will lead to new questions and challenges, both on the technical side as well as on a human level. Within this work, the social behavior of the robots was restricted to a superficial level. On the software side, it will be interesting to see how social interaction algorithms can be made accessible to non-experts, so that a wide audience of users can create deep social behaviors for their robotic creations.

The DIY toolkit approach that was used during this research project holds much potential for a broad range of scientific disciplines. We found this paradigm especially valuable for many reasons. To begin, the approach has a leverage effect: a lot more ground can be covered by enabling motivated lead users to help. Related to this is that toolkits offer new pathways of knowledge dissemination. It enabling the public to experiment with cutting edge technology, and helps new developments to “escape” the lab. Finally, we found that constraints imposed by the DIY mindset stimulated creativity not only in users but also in ourselves, stimulating us to explore paths we otherwise would not have considered.

## 5.1 OPEN HARDWARE IN ROBOTICS RESEARCH

Implementing the open source hardware paradigm in robotics research involves a number of ethical and practical considerations. From a moral standpoint, it stands to reason that the results of government-funded research – not only publications, but also research platforms – should be made publicly available. After all, the government, the employer of most researchers, represents the public at large. Doing so fits within the scope of knowledge dissemination, one of the core tasks of a university. Furthermore, releasing the mechanical designs involved in a publication as open source hardware can be likened to releasing an experiment’s dataset, and serves to improve third-party verification of the research.

There are also practical reasons to release research hardware as an open source work. If others can freely use and modify the work, they may also be inclined to improve upon it and release their changes in a similar manner. This collaborative process has much poten-

tial, though in our experience this can be hit-or-miss. The source files of the robot Ono, the precursor to the Opsoro platform, have been made available for a number of years now, though we failed to attract outside collaborators. A successful open source hardware project requires more than merely making source files available. Documentation, training, and community management also take considerable effort, for which there is not always time in a research context. Other contributing factors are that hardware is naturally harder to replicate than software, and that hardware created for research tends to be very specific, resulting in a niche of potential users. More recently, we have had some success in attracting outside collaborators for the Opsoro platform. This is mainly the result of addressing community-related factors, including organizing workshops, improving documentation, offering better software, and improving communication through social media. However, researchers often face the challenge to find time for activities that do not directly lead to publishable results. To go from an experimental setup to a viable open source hardware project requires a significant efforts in development and documentation work.

In recent years, ready-to-use platforms (such as Nao, Gouaillier et al., 2008) have had a large impact on HRI research. On the other hand, developing custom robotic platforms, though costly, remains a common approach as well. However, both approaches are often technology-driven and difficult to modify, and little exists in between these extremes. With our work toward the Opsoro platform, we address this gap by combining elements of off-the-shelf robots and custom-designed robots. Using the platform is easier, faster and cheaper than building costly custom robots. And because the platform is open, users are free to modify any and every aspect as they see fit, which is often the case in scientific experiments. While the open source hardware could drive down the costs of robotics, we find that many existing open source robotics platforms are still expensive primarily because of component costs. This cost poses a barrier for replication and modification, hampering the evolutionary process behind open source projects. Our work explicitly takes this into account by relying on low-cost components and techniques. Instead of offering advanced capabilities, we rely on a bottom-up approach, focused on offering a strong basis of basic functionality for others to build upon.

The field of robotics is very challenging and reliant on many different disciplines. An advantage of the open source approach is that users are free to cherry-pick the best tools and components in order to create a larger, functional device. Designers do not need to know all the intricate details of every component, they just need to know how to “glue” everything together. This is in itself a very powerful method for technology democratization. The open source hardware movement is very young, but has already made a measurable impact on society. Still, challenges remain; DIY researchers and practitioners stand much to gain from better tools for collaboration, for documentation, for manufacturing, and for design. It will be interesting to see what happens when the DIY world and the world of academia move closer together, as they both have much to gain from each other.

## 5.2 ENTREPRENEURSHIP AND VALORIZATION

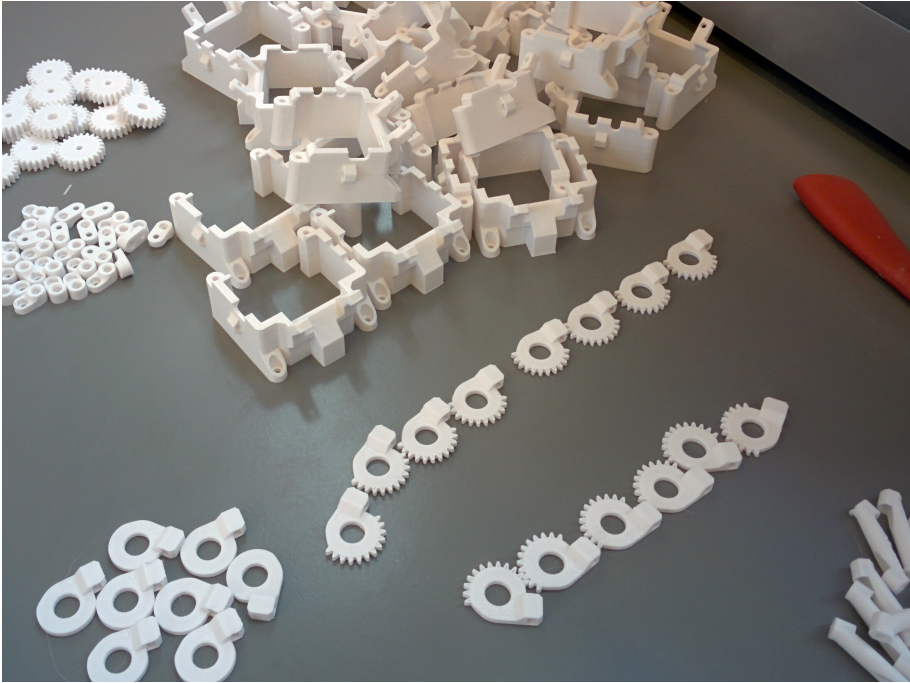
Owing to our background in industrial design engineering, we have employed a practical approach during this project, paying attention to aspects of industrialization, production, and entrepreneurship. As a study within the industrial sciences, relevance to industry is an important factor of the finality of this work. As discussed in the previous section, the success of the Opsoro platform requires both research and development work. However, pure development and implementation is often difficult in a university environment. To enable the success of the platform, we are moving forward with Opsoro in the form of a start-up company. Pioneering companies, such as Arduino and Ultimaker, have demonstrated that it is possible to be both open source *and* profitable. However, a key challenge is to figure out a viable open source business model, which tends to be unique to each individual case. Still, we remain convinced that the open source approach offers many more opportunities for Opsoro than risks.

Over the course of this research project, we often went one step further than bare minimum proof-of-concept prototypes, employing an integral product design philosophy that is uncharacteristic for robotics research. More often than not, this worked out to save time overall, as the extra time spent on a robust implementation was usually made up by the fact that future developments could be implemented more quickly. For instance, the time we spent to design a custom PCB was ultimately justified because of the increased reliability and better scalability. It let us replicate prototypes faster, enabling larger experiments and ultimately also robot sales.

The attention for production is also true for the 3D printed parts. They were designed and printed on low-cost machines, even though our department has access to much better high-end machines. Doing so, we could better optimize the designs for low-end 3D printers, and ultimately enabled us to do a small “production” run on our eight DIY-grade machines (fig. 5.3). In some respects, the Opsoro toolkit leans more toward product prototype rather than research prototype, illustrating the point that this doctoral research project originated in industrial design.

Entrepreneurship and valorization is also shown in our participation in outreach events, where we communicate our work outside of the traditional academic channels. Ono was presented at Maker Faire Rome in 2013, where it won the *Maker of Merit* award. We have also been present at every edition of Maker Faire Mechelen since the first edition in 2014. Since then, Ono and Opsoro have been appeared multiple times in Belgian popular media, including national television, radio, and newspapers. Response to these outreach events, feedback at conferences, sales of prototype robots to other universities made it clear that the Opsoro platform has potential for commercialization as a spin-off company. In late 2015, we submitted a proposal for funding to valorize our research in the form of a start-up company. Our proposal was granted, giving us the opportunity to make the necessary preparations for a spin-off company.

Three potential markets for Opsoro were identified: (1) products for robot-assisted therapy and care, (2) tools for human-robot interaction research, and (3) kits for STEAM



**Fig. 5.3** Opsoro components produced by a farm of eight low-cost 3D printers.

education. Through interviews and further market analysis it became apparent that the first two markets were not (yet) feasible for us. Major issues in the therapeutic market include device robustness, customer support, medical certification. The largest problem with the research market is simply the limited size of this market, making it difficult to offset the costs of research, development, tooling and manufacture. Consequently, primary focus for the Opsoro product will be on the hobbyist market, emphasizing STEAM learning. As of yet, this market is dominated by male-oriented products, such as the popular LEGO Mindstorms toy series. Within this market, we are in a unique position because we can offer a robot building kit that emphasizes aspects that are often overlooked by other products in favor of cool, mechanical, technical, aggressive product aesthetics, which tend to resonate much more with boys than with girls. With Opsoro as a basis, we can offer a product that plays into aspects such as storytelling, social interaction, creature design, and creative expression, thus creating a product that frames STEAM skills in the fantasy world of Disney and Pixar.

This change in direction has repercussions for the development of the product. Most importantly, the cost needs to be much lower in order to have a viable customer value proposition. The price of a Mindstorms kit – roughly 400 € – seems to be the maximum the market can bear. One proposed change is to eliminate textiles from the design of the kit in favor of other materials. The production of a textile skin involves a lot of manual labor, which is expensive. Furthermore, by offering the product as a kit, some of the assembly



costs are eliminated. Design-wise, one of the challenges will be to bridge the differences between mass-production manufacturing and DIY-centric manufacturing. We want to design our components so that they can be manufactured at home by end-users, but can also be produced efficiently in large volumes. We think it is important to emphasize Maker & DIY skills as part of the design process. Instead of designing a system that is restricted to components specific to that system, the system should be extensible. The challenge here is to find different ways of combining craft techniques with standardized toolkit components, allowing users to build their own embodiments with a high degree of flexibility.

Creating a business around an open source product has deep implications for the product and the company. Most importantly, an open source product should benefit the users, allowing them to easily hack functionality outside of the scope of the original product. On the other hand, there should be clear advantages for buying an Opsoro kit versus making or buying a knock-off copy. Here, multiple strategies can be used to create value. For instance, a commercial Opsoro kit can be faster and of better quality than a home-made kit. Still, one of the greatest assets of an open source company is its community, and this community cannot be copied by third-party manufacturers. For this reason, the online community platform will be one of the next big steps toward commercialization.

The platform approach ties in well with our broader vision on advances within technology. Toolkits and platforms make it possible for end users to develop their own robotics applications, and this approach is completely different from the technology push approach of many contemporary robotics companies. This approach offers a better way to disseminate research knowledge and to enhance the impact of the work. The link between the research community and makers and DIYers offers a great way to put research into practice. The approach also offers advantages from a business perspective. If we succeed in fostering a community around the online platform and the physical toolkit, we will be in a unique position to identify new trends within HRI applications. As a first step, the spin-off company will offer an open-ended platform for DIY social robots. At a later stage, we can take advantage of insights gained through the platform by offering more specialized products and services.

As technology continues to permeate throughout all aspects of our daily lives, being able to use technology as a creative medium becomes an increasingly important life skill. DIY paradigms offer a promising solution to bridge the gap between the research world and society. Social robotics is but one example of advances in technology, though it is currently the subject of much attention and the technology will surely affect the shape of future societies. Throughout the past four years, we have realized much in terms of democratizing social robotics technology. With the prospect of starting a spin-off company<sup>2</sup>, it is safe to say the future will hold many more exciting challenges and opportunities.

<sup>2</sup>More information on the spin-off company can be found on the website <http://www.opsoro.be>.



# BIBLIOGRAPHY

- Abel, B. van, L. Evers, P. Troxler, and R. Klaassen, eds. (2011). *Open Design Now: Why Design Cannot Remain Exclusive*. Amsterdam: BIS publishers, p. 256.
- Als, B. S., J. J. Jensen, and M. B. Skov (2005). “Comparison of think-aloud and constructive interaction in usability testing with children”. In: *Proceedings of the 2005 conference on Interaction design and children*. New York: ACM Press, pp. 9–16.
- Altendorfer, R., N. Moore, H. Komsuoglu, M. Buehler, H. B. Brown, D. McMordie, U. Saranlı, R. Full, and D. E. Koditschek (2001). “RHex: A biologically inspired hexapod runner”. In: *Autonomous Robots* 11.3, pp. 207–213.
- American Psychiatric Association (2013). *Diagnostic and Statistical Manual of Mental Disorders (DSM-5)*. 5th editio, p. 947.
- Anderson, C. (2008). *The Long Tail: How Endless Choice Is Creating Unlimited Demand*. Revised ed. Hyperion, p. 256.
- (2012). *Makers: The new industrial revolution*. Crown Business.
- Araújo, A., D. Portugal, M. S. Couceiro, and R. P. Rocha (2014). “Integrating Arduino-Based Educational Mobile Robots in ROS”. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 77.2, pp. 281–298.
- Baafi, E. (2014). “Standardization of Open Source Hardware”. In: *Building Open Source Hardware: DIY Manufacturing for Hackers and Makers*.
- Baird, G., E. Simonoff, A. Pickles, S. Chandler, T. Loucas, D. Meldrum, and T. Charman (2006). “Prevalence of disorders of the autism spectrum in a population cohort of children in South Thames: the Special Needs and Autism Project (SNAP)”. In: *Lancet* 368.9531, pp. 210–215.
- Bangor, A., P. T. Kortum, and J. T. Miller (2008). “An Empirical Evaluation of the System Usability Scale”. In: *International Journal of Human-Computer Interaction* 24.6, pp. 574–594.
- Barragán, H. (2004). “Wiring: Prototyping physical interaction design.” PhD thesis. Interaction Design Institute Ivrea.
- Bartneck, C. and J. Forlizzi (2004). “A design-centred framework for social human-robot interaction”. In: *Proceedings of the 13<sup>th</sup> IEEE International Workshop on Robot and Human Interactive Communication*. IEEE, pp. 591–594.
- Bartneck, C., J. Reichenbach, and A. Breemen (2004). “In your face, robot! The influence of a character’s embodiment on how users perceive its emotional expressions”. In: *Proceedings of the 4<sup>th</sup> Design and Emotion Conference*. Ankara.

- Bdeir, A. (2011). "Electronics as Material : littleBits". In: *Proceedings of the 3<sup>rd</sup> International Conference on Tangible and Embedded Interaction*, pp. 3–6.
- Beck, K., M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, and R. Jeffries (2001). *Manifesto for agile software development*.
- Beer, R. D., R. D. Quinn, H. J. Chiel, and R. E. Ritzmann (1997). "Biologically Inspired Approaches to Robotics". In: *Communications of the ACM* 40.3, pp. 31–38.
- Bell, S. (2010). "Project-based learning for the 21st century: Skills for the future". In: *The Clearing House: A Journal of Educational Strategies, Issues and Ideas* 83.2, pp. 39–43.
- Benitti, F. B. V. (2012). "Exploring the educational potential of robotics in schools: A systematic review". In: *Computers & Education* 58.3, pp. 978–988.
- Bequette, J. W. and M. B. Bequette (2012). "A Place for ART and DESIGN Education in the STEM Conversation". In: *Art Education* March, pp. 40–47.
- Biggs, J. (2015). *Makerbot's Saddest Hour*. URL: <https://techcrunch.com/2015/04/22/makerbots-saddest-hour/>.
- Birkmeyer, P., K. Peterson, and R. S. Fearing (2009). "DASH: A dynamic 16g hexapedal robot". In: *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2683–2689.
- Blank, D., D. Kumar, L. Meeden, and H. Yanco (2003). "Pyro: A python-based versatile programming environment for teaching robotics". In: *ACM Journal of Educational Resources in Computing* 3.4, pp. 1–27.
- Blikstein, P. (2013a). "Digital Fabrication and "Making" in Education: The Democratization of Invention". In: *FabLabs: Of Machines, Makers and Inventors*. Ed. by J. Walter-Herrmann and C. Büching. Bielefeld: Transcript-Verlag.
- Blikstein, P. (2013b). "Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future". In: *Proceedings of the 12<sup>th</sup> International Conference on Interaction Design and Children*, pp. 173–182.
- Bouchard, D., V. St, S. Daniels, and V. St (2015). "Tiles that Talk : Tangible Templates for Networked Objects". In: *Proceedings of the 9<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*, pp. 197–200.
- Brandt, A. M. and M. B. Colton (2008). "Toys in the classroom: LEGO MindStorms as an educational haptics platform". In: *Proceedings of the 2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. IEEE, pp. 389–395.
- Breazeal, C. (2003a). *Designing sociable robots*. Vol. 45. MIT Press, p. 1774.
- (2003b). "Emotion and sociable humanoid robots". In: *International Journal of Human Computer Studies* 59.1-2, pp. 119–155.
- (2003c). "Toward sociable robots". In: *Robotics and Autonomous Systems* 42.3-4, pp. 167–175.
- Brooke, J. (1996). "SUS-A quick and dirty usability scale". In: *Usability evaluation in industry*. Ed. by P. W. Jordan, B. Thomas, B. Weerdmeester, and I. L. McClelland. London: CRC Press. Chap. 21, pp. 189–194.
- Brown, T. (2008). "Design thinking". In: *Harvard Business Review* 86.6, pp. 84–92.
- Buchanan, R. (1992). "Wicked problems in design thinking". In: *Design Issues* 8.2, pp. 5–21.

- Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design: Getting the Design Right and the Right Design*. Amsterdam: Morgan Kaufmann, p. 443.
- Cabibihan, J.-J., H. Javed, M. J. Ang, and S. M. Aljunied (2013). "Why robots? A survey on the roles and benefits of social robots in the therapy of children with autism". In: *International Journal of Social Robotics* 5.4, pp. 593–618.
- Cañamero, L. D. and J. Fredslund (2000). "How Does It Feel? Emotional Interaction with a Humanoid LEGO Robot". In: *Proceedings of the 2000 Association for the Advancement of Artificial Intelligence Fall Symposium*, pp. 23–28.
- Cao, H. L., G. Van De Perre, R. Simut, C. Pop, A. Peca, D. Lefebber, and B. Vanderborght (2014). "Enhancing My Keepon robot: A simple and low-cost solution for robot platform in Human-Robot Interaction studies". In: *Proceedings of the The 23<sup>rd</sup> IEEE International Symposium on Robot and Human Interactive Communication*. Edinburgh, UK, pp. 555–560.
- Capraro, R. M., M. M. Capraro, and J. R. Morgan (2013). *STEM Project-Based Learning*. Rotterdam: Sense Publishers.
- Carbajal, J. P., D. Assaf, and E. Benker (2011). "Promoting scientific thinking with robots". In: *Proceedings of the 2<sup>nd</sup> International Conference on Robotics in Education*, pp. 59–61.
- Ceruzzi, P. E. (2003). *A History of Modern Computing*. Second edi. MIT Press, p. 459.
- Chaiklin, S. (2003). "The Zone of Proximal Development in Vygotsky's Analysis of Learning and Instruction". In: *Vygotsky's Educational Theory in Cultural Context*. Ed. by A. Kozulin, B. Gindis, V. S. Ageyev, and S. M. Miller. Cambridge University Press, pp. 39–64.
- Chung, J., K. Min, and W. Lee (2013). "CUBEMENT: Democratizing Mechanical Movement Design". In: *Proceedings of the 8<sup>th</sup> International Conference on Tangible Embedded and Embodied Interaction*, pp. 81–84.
- Church, C. W. (1967). "Wicked Problems". In: *Management Science* 14.4.
- Church, W., T. Ford, N. Perova, and C. Rogers (2010). "Physics With Robotics-Using LEGO MINDSTORMS In High School Education." In: *Proceedings of the 2010 Association for the Advancement of Artificial Intelligence Spring Symposium*.
- Cole, J. (1998). *About Face*. Cambridge, Massachusetts: MIT Press, p. 237.
- Conradie, P. D., C. Vandevelde, J. De Ville, and J. Saldien (2016). "Prototyping Tangible User Interfaces: Case Study of the Collaboration between Academia and Industry". In: *International Journal of Engineering Education* 32.2, pp. 1–12.
- Dabbish, L., C. Stuart, J. Tsay, and J. Herbsleb (2012). "Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository". In: *Proceedings of the 2012 ACM Conference on Computer Supported Cooperative Work*, pp. 1277–1286.
- Dautenhahn, K., C. L. Nehaniv, M. L. Walters, B. Robins, H. Kose-Bagci, N. A. Mirza, and M. Blow (2009). "KASPAR – a minimally expressive humanoid robot for human–robot interaction research". In: *Applied Bionics and Biomechanics* 6.3–4, pp. 369–397.
- De Beir, A., H.-I. Cao, P. Gomez, G. Van De Perre, and B. Vanderborght (2015). "Enhancing Nao Expression of Emotions Using Pluggable Eyebrows". In: *Proceedings of the 1<sup>st</sup> International Conference on Social Robots in Therapy and Education*. Almere, pp. 36–38.

- De Beir, A. and B. Vanderborght (2016). "Evolutionary Method for Robot Morphology : Case Study of Social Robot Probo". In: *Proceedings of the 11<sup>th</sup> ACM/IEEE International Conference on Human Robot Interaction*. Christchurch, New Zealand, pp. 609–610.
- Design Council (2005). *A Study of the Design Process*. Tech. rep., p. 144.
- Desmet, P., M. Vastenburg, D. Van Bel, and N. Romero (2012). "Pick-A-Mood - Development and Application of a Pictorial Mood Reporting Instrument". In: *Proceedings of the 8<sup>th</sup> International Design and Emotion Conference*.
- Dietz, P. H., G. Reyes, and D. Kim (2014). "The PumpSpark fountain development kit". In: *Proceedings of the 2014 International Conference on Designing Interactive Systems*. New York, New York, USA: ACM Press, pp. 259–266.
- Disalvo, C. F., F. Gemperle, J. Forlizzi, and S. Kiesler (2002). "All Robots Are Not Created Equal: The Design and Perception of Humanoid Robot Heads". In: *Proceedings of the 4<sup>th</sup> conference on designing interactive systems: processes, practices, methods, and techniques*, pp. 321–326.
- Doroftei, I., F. Adascalitei, D. Lefebvre, B. Vanderborght, and I. A. Doroftei (2016). "Facial expressions recognition with an emotion expressive robotic head". In: *Proceedings of the 7<sup>th</sup> International Conference on Advanced Concepts in Mechanical Engineering*, p. 147.
- Dorst, K. (2010). "The Nature of Design Thinking". In: *Proceedings of the 8<sup>th</sup> Design Thinking Research Symposium*, pp. 131–139.
- Dougherty, D. (2008). "The Joy of Making". In: *Proceedings of the 2<sup>nd</sup> IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*. IEEE, pp. 8–12.
- (2012). "The maker movement". In: *Innovations* 7.3, pp. 11–14.
- Draper, S. W. (1999). "Analysing fun as a candidate software requirement". In: *Personal Technologies* 3.3, pp. 117–122.
- Duffy, B. (2003). "Anthropomorphism and the social robot". In: *Robotics and autonomous systems* 42.3-4, pp. 177–190.
- Eastham, B. (2015). "Panasonic and the OpenDOF Project : Open-source vision in a large company". In: *Crossroads* 22.2, pp. 58–61.
- Ekman, P. (1992). "Are there basic emotions?" In: *Psychological Review* 99.3, pp. 550–553.
- (1999). "Basic emotions". In: *Handbook of cognition and emotion*. Ed. by T. Dalgleish and M. Power. Vol. 98. 1992. New York: Wiley, pp. 45–60.
- Ekman, P., W. Friesen, and J. Hager (1978). *Facial Action Coding System: A technique for the measurement of facial action*. Consulting Psychologists Press.
- EU (2000). *Presidency conclusions - Lisbon European Council 23 and 24 march 2000*. URL: [http://www.consilium.europa.eu/en/uedocs/cms\\_data/docs/pressdata/en/ec/00100-r1.en0.htm](http://www.consilium.europa.eu/en/uedocs/cms_data/docs/pressdata/en/ec/00100-r1.en0.htm).
- (2010). *Europe 2020: A strategy for smart, sustainable and inclusive growth*. Tech. rep. Brussels.
- Feisel, L. D. and A. J. Rosa (2005). "The role of the laboratory in staff development." In: *Journal of Engineering Education* 94.1, pp. 121–130.
- Fischer, G. (1998). "Beyond "Couch Potatoes": From Consumers to Designers and Active Contributors". In: *Proceedings of the 3<sup>rd</sup> Asia Pacific Conference on Computer Human Interaction*, pp. 1–30.
- Fisher, R. (1922). "On the interpretation of  $\chi^2$  from contingency tables, and the calculation of P". In: *Journal of the Royal Statistical Society* 85.1, pp. 87–94.

- Fong, T., I. Nourbakhsh, and K. Dautenhahn (2003). "A survey of socially interactive robots". In: *Robotics and Autonomous Systems* 42.3-4, pp. 143-166.
- Fonteyn, M., B. Kuipers, and S. Grobe (1993). "A description of think aloud method and protocol analysis". In: *Qualitative Health Research* 3.4, pp. 430-441.
- Fornari, D. and S. Cangiano (2015). "Open Sourcing Social Robotics: Humanoid Artifacts from the Viewpoint of Designers". In: *Social Robots from a Human Perspective*. Ed. by J. Vincent, S. Taipale, B. Sapio, G. Lugano, and L. Fortunati, pp. 98-102.
- Forsslund, J., M. C. Yip, and E.-L. Sallnas (2015). "WoodenHaptics : A Starting Kit for Crafting Force-Reflecting Spatial Haptic Devices". In: *Proceedings of the 9<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*, pp. 133-140.
- Fortus, D. and J. Krajcik (2005). "Design-based science and real-world problem-solving". In: *International Journal of Science Education* 27.7, pp. 855-879.
- Gates, B. (2007). "A robot in every home." In: *Scientific American* 296.1, pp. 58-65.
- Gershenfeld, N. (2012). "How to Make Almost Anything: The Digital Fabrication Revolution". In: *Foreign Affairs* 91.6, pp. 43-57.
- Gershenfeld, N. A. (2005). *Fab: the coming revolution on your desktop - from personal computers to personal fabrication*. Basic Books.
- Gibb, A. (2014). *Building Open Source Hardware: DIY Manufacturing for Hackers and Makers*. Addison-Wesley Professional, p. 368.
- Goetz, J., S. Kiesler, and A. Powers (2003). "Matching robot appearance and behavior to tasks to improve human-robot cooperation". In: *Proceedings of the 2003 IEEE International Workshop on Robot and Human Interactive Communication*, pp. 55-60.
- Goldstein, C. (1998). *Do It Yourself: Home Improvement in 20th-Century America*. Ed. by S. E. Stemen. First edit. New York: Princeton Architectural Press, p. 120.
- Gonzalez-Gomez, J., A. Valero-Gomez, A. Prieto-Moreno, and M. Abderrahim (2012). "A New Open Source 3D-Printable Mobile Robotic Platform for Education". In: *Advances in Autonomous Mini Robots*. Ed. by U. Rückert, S. Joaquin, and W. Felix. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 49-62.
- Goris, K., J. Saldien, B. Vanderborght, and D. Lefeber (2011). "How to achieve the huggable behavior of the social robot Probo? A reflection on the actuators". In: *Mechatronics* 21.3, pp. 490-500.
- Gouaillier, D., V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier (2008). "The NAO humanoid: a combination of performance and affordability". In: *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pp. 1-10.
- Goud, R., S. P. Krishnamoorthy, and V. Kapila (2015). "A Blocks-based Visual Environment to Teach Robot-Programming to K-12 Students". In: *122<sup>nd</sup> ASEE Annual Conference & Exposition*, pp. 1-12.
- Grassé, P.-P. (1959). "La reconstruction du nid et les coordinations interindividuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. La théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs". In: *Insectes Sociaux* 6.1, pp. 41-80.
- Guizzo, E. and E. Ackerman (2012). "The rise of the robot worker". In: *IEEE Spectrum* 49.10, pp. 34-41.
- Guizzo, E. and T. Deyle (2012). "Robotics trends for 2012". In: *IEEE Robotics and Automation Magazine* 19.1, pp. 119-123.

- Ha, I., Y. Tamura, H. Asama, J. Han, and D. W. Hong (2011). "Development of open humanoid platform DARwIn-OP". In: *Proceedings of the 2011 Annual Conference of the Society of Instrument and Control Engineers*, pp. 2178–2181.
- Hake, R. (1998). "Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses". In: *American journal of Physics* 66.1, pp. 64–74.
- Haring, K. (2003). "The "Freer Men" of Ham Radio". In: *Technology and Culture* 44.4, pp. 734–761.
- Hartmann, B., S. Doorley, and S. R. Klemmer (2008). "Hacking, mashing, gluing: Understanding opportunistic design". In: *IEEE Pervasive Computing* 7.3, pp. 46–54.
- Hartmann, S., H. Wiesner, and A. Wiesner-Steiner (2007). "Robotics and gender: The use of robotics for the empowerment of girls in the classroom". In: *Gender Designs IT*, pp. 175–188.
- Hassenzahl, M. (2003). "The Thing and I: Understanding the Relationship Between User and Product". In: *Funology: From Usability to Enjoyment*. Ed. by M. A. Blythe, K. Overbeeke, A. F. Monk, and P. C. Wright. Springer Netherlands, pp. 31–42.
- Hassenzahl, M., M. Burmester, and F. Koller (2003). "AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität". In: *Mensch & Computer*. Ed. by G. Szwillus and J. Ziegler. Vol. 57. Berichte des German Chapter of the ACM. Wiesbaden: Vieweg+Teubner Verlag, pp. 187–196.
- Hatch, M. (2013). *The Maker Movement Manifesto: Rules for Innovation in the New World of Crafters, Hackers, and Tinkerers*. McGraw-Hill Professional, p. 256.
- Herrmann, K. H., C. Gärtner, D. Güllmar, M. Krämer, and J. R. Reichenbach (2014). "3D printing of MRI compatible components: Why every MRI research group should have a low-budget 3D printer". In: *Medical Engineering and Physics* 36.10, pp. 1373–1380.
- Hippel, E. von (2001). "PERSPECTIVE: User toolkits for innovation". In: *Journal of Product Innovation Management* 18.4, pp. 247–257.
- Hippel, E. von and R. Katz (2002). "Shifting Innovation to Users via Toolkits". In: *Management Science* 48.7, pp. 821–833.
- Hippel, E. von and G. von Krogh (2003). "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science". In: *Organization Science* 14.2, pp. 209–223.
- Hoffman, G. (2012). "Dumb robots, smart phones: A case study of music listening companionship". In: *Proceedings of the 2012 IEEE International Workshop on Robot and Human Interactive Communication*, pp. 358–363.
- Honey, M. and D. E. Kanter (2013). *Design, Make, Play: Growing the next generation of STEM innovators*. Routledge, p. 256.
- Ierusalimsky, R., L. H. de Figueiredo, and W. Celes (2007). "The Evolution of Lua". In: *Proceedings of the 3<sup>rd</sup> ACM SIGPLAN conference on History of programming languages*. San Diego, California: ACM, pp. 2–1–2–26.
- IFR (2016a). *Executive Summary of World Robotics 2016 Industrial Robots*. Tech. rep. International Federation of Robotics, p. 8.
- (2016b). *Executive Summary of World Robotics 2016 Service Robots*. Tech. rep. International Federation of Robotics, p. 8.



- Irwin, J. L., D. E. Oppliger, J. M. Pearce, and G. Anzalone (2015). "Evaluation of RepRap 3D Printer Workshops in K-12 STEM". In: *Proceedings of the 2015 ASEE Annual Conference and Exposition*, pp. 26.696.1–26.696.18.
- Ishii, H. (2008). "Tangible bits: beyond pixels". In: *Proceedings of the 2<sup>nd</sup> international conference on Tangible and Embedded Interaction*, pp. xv–xxv.
- Ishii, H. and B. Ullmer (1997). "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms". In: *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pp. 234–241.
- ISO (1998). *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs): guidance on usability*. Geneva.
- (2010). *ISO 9241-210:2010: Human-centred design for interactive systems*. Geneva.
- (2012). *ISO 8373:2012: Robots and robotic devices - Vocabulary*. Geneva.
- Johnson, J. (2003). "Children, robotics, and education". In: *Artificial Life and Robotics* 7.1-2, pp. 16–21.
- Johnston, O. and F. Thomas (1995). *The Illusion of Life: Disney Animation*. 2nd edition. New York: Hyperion, p. 576.
- Jones, R., P. Haufe, E. Sells, P. Iravani, V. Olliver, C. Palmer, and A. Bowyer (2011). "RepRap – the replicating rapid prototyper". In: *Robotica* 29.Special Issue 01, pp. 177–191.
- Jordan, P. W. (2000). *Designing pleasurable products: an introduction to the new human factors*. CRC Press, p. 224.
- Juang, H. S. and K. Y. Lurr (2013). "Design and control of a two-wheel self-balancing robot using the arduino microcontroller board". In: *Proceedings of the IEEE International Conference on Control and Automation*, pp. 634–639.
- Kahneman, D. (2013). *Thinking Fast and Slow*. Farrar, Straus and Giroux, p. 499.
- Kaya, N. and H. H. Epps (2004). "Relationship between color and emotion: a study of college students". In: *College Student Journal* 38.3, pp. 396–405.
- Kedzierski, J., R. Muszyński, C. Zoll, A. Oleksy, and M. Frontkiewicz (2013). "EMYS-Emotive Head of a Social Robot". In: *International Journal of Social Robotics* 5.2, pp. 237–249.
- Kemp, J. and T. van Gelderen (1996). "Co-discovery exploration: an informal method for the iterative design of consumer products". In: *Usability Evaluation In Industry*. Ed. by P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester. CRC Press. Chap. 16, p. 252.
- Kesteren, I. E. H. van, M. M. Bekker, A. P. O. S. Vermeeren, and P. A. Lloyd (2003). "Assessing usability evaluation methods on their effectiveness to elicit verbal comments from children subjects". In: *Proceedings of the 2003 Conference on Interaction Design and Children*. New York: ACM Press, p. 41.
- Kidd, C. D., W. Taggart, and S. Turkle (2006). "A sociable robot to encourage social interaction among the elderly". In: *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 3972–3976.
- Kipp, A. and S. Schneider (2016). "Applied Social Robotics – Building Interactive Robots with LEGO Mindstorms". In: *Robotics in Education: Research and Practices for Robotics in STEM Education*. Ed. by M. Merdan, W. Lepuschitz, G. Koppensteiner, and R. Balogh. Springer International Publishing, pp. 29–40.

- Koenig, N. and A. Howard (2004). "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai, Japan, pp. 2149–2154.
- Kolko, J. (2010). "Abductive Thinking and Sensemaking: The Drivers of Design Synthesis". In: *Design Issues* 26.1, pp. 15–28.
- Kolodner, J. and P. Camp (2003). "Problem-based learning meets case-based reasoning in the middle-school science classroom: Putting learning by design (tm) into practice". In: *Journal of the Learning Sciences* 12.4.
- Kozima, H., M. P. Michalowski, and C. Nakagawa (2009). "Keepon: A Playful Robot for Research, Therapy, and Entertainment". In: *International Journal of Social Robotics* 1.1, pp. 3–18.
- Kozima, H., C. Nakagawa, and Y. Yasuda (2005). "Interactive robots for communication-care: A case-study in autism therapy". In: *Proceedings of the 2005 IEEE International Workshop on Robot and Human Interactive Communication*, pp. 341–346.
- Krishna, G. (2015). *The Best Interface Is No Interface: The simple path to brilliant technology*. Ed. by M. Nolan, N. Peterson, B. Lindstrom, and D. Meiss. New Riders, p. 256.
- Kujala, S., V. Roto, K. Vaananen-Vainio-Mattila, E. Karapanos, and A. Sinnela (2011). "UX Curve: A method for evaluating long-term user experience". In: *Interacting with Computers* 23.5, pp. 473–483.
- Kuznetsov, S. and E. Paulos (2010). "Rise of the expert amateur: DIY projects, communities, and cultures". In: *Proceedings of the 6<sup>th</sup> Nordic Conference on Human-Computer Interaction*, pp. 295–304.
- Lapeyre, M., P. Rouanet, J. Grizou, S. Nguyen, F. Depraetre, A. Le Falher, and P.-y. Oudeyer (2014). "Poppy Project: Open-Source Fabrication of 3D Printed Humanoid Robot for Science, Education and Art". In: *Proceedings of the Digital Intelligence Conference 2014*. Nantes, p. 6.
- Lee, K. M., Y. Jung, J. Kim, and S. R. Kim (2006). "Are physically embodied social agents better than disembodied social agents?: The effects of physical embodiment, tactile interaction, and people's loneliness in human-robot interaction". In: *International Journal of Human Computer Studies* 64.10, pp. 962–973.
- Leite, I., C. Martinho, and A. Paiva (2013). "Social Robots for Long-Term Interaction: A Survey". In: *International Journal of Social Robotics* 5.2, pp. 291–308.
- Leite, I., A. A. Pereira, C. Martinho, and A. Paiva (2008). "Are emotional robots more fun to play with?" In: *Proceedings of the 17<sup>th</sup> IEEE International Symposium on Robot and Human Interactive Communication*, pp. 77–82.
- Lenhart, A. and M. Madden (2005). *Teen Content Creators and Consumers*. Tech. rep., p. 29.
- Li, D., P. L. P. Rau, and Y. Li (2010). "A cross-cultural study: Effect of robot appearance and task". In: *International Journal of Social Robotics* 2.2, pp. 175–186.
- Lipson, H. and M. Kurman (2013). *Fabricated: The New World of 3D Printing*. Wiley.
- Ma, R. R., L. U. Odhner, and A. M. Dollar (2013). "A modular, open-source 3D printed underactuated hand". In: *Proceedings of the 2013 IEEE International Conference on Robotics and Automation*, pp. 2737–2743.
- MacDorman, K. F. (2005). "Androids as an experimental apparatus: Why is there an uncanny valley and can we exploit it". In: *Proceedings of the 2005 Cognitive Science Society - Workshop: Toward Social Mechanisms of Android Science*, pp. 106–118.

- Maeda, J. (2013). "STEM + Art = STEAM". In: *The STEAM Journal* 1.1, pp. 1–3.
- Martin, L. (2015). "The Promise of the Maker Movement for Education". In: *Journal of Pre-College Engineering Education Research* 5.1, pp. 1–30.
- Maslow, A. (1943). "A theory of human motivation". In: *Psychological Review* 50, pp. 370–396.
- Mason, P. (2016). *BrewDog's open-source revolution is at the vanguard of postcapitalism*. URL: <https://www.theguardian.com/commentisfree/2016/feb/29/brewdogs-open-source-revolution-is-at-the-vanguard-of-postcapitalism> (visited on 12/05/2016).
- Massie, K. and S. D. Perry (2002). "Hugo Gernsback and radio magazines: An influential intersection in broadcast history". In: *Journal of Radio Studies* 9.2, pp. 264–281.
- Mataric, M. J., N. P. Koenig, and D. Feil-Seifer (2007). "Materials for Enabling Hands-On Robotics and STEM Education." In: *Proceedings of the 2007 Association for the Advancement of Artificial Intelligence Spring Symposium*, pp. 99–102.
- McPherson, S. (2014). "Strategies and Resources for Preparing Teachers for STEM Teaching and Learning". In: *Proceedings of the 2014 International Conference of the Society for Information Technology & Teacher Education*, pp. 1927–1939.
- Megaro, V., B. Thomaszewski, and M. Nitti (2015). "Interactive design of 3D-printable robotic creatures". In: *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH Asia 2015* 34.6.
- Mehrabian, A. (1995). "Framework for a comprehensive description and measurement of emotional states." In: *Genetic, social, and general psychology monographs* 121.1978, pp. 339–361.
- Mehrabian, A. (2008). "Communication without words." In: *Communication Theory*. Ed. by C. D. Mortensen. Second edi. New Brunswick: Transaction Publishers, pp. 193–200.
- Mellis, D. A. (2014). "Personal Manufacturing in the Digital Age". In: *Building Open Source Hardware: DIY Manufacturing for Hackers and Makers*, pp. 149–160.
- Mellis, D. A. and L. Buechley (2014). "Do-it-yourself Cellphones: An Investigation into the Possibilities and Limits of High-tech Diy". In: *Proceedings of the 2014 SIGCHI Conference on Human Factors in Computing Systems*, pp. 1723–1732.
- Mellis, D. A., T. Igoe, M. Banzi, and D. Cuartielles (2007). "Arduino: An open electronic prototyping platform". In: *Proceedings of the 2007 SIGCHI Conference on Human Factors in Computing Systems*. San Jose, USA: ACM Press.
- Mellis, D. A., S. Jacoby, L. Buechley, H. Perner-wilson, and J. Qi (2013). "Microcontrollers as material: crafting circuits with paper, conductive ink, electronic components, and an untookit". In: *Proceedings of the 7<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*, pp. 83–90.
- Mellis, D. and L. Buechley (2012). "Collaboration in open-source hardware: third-party variations on the arduino duemilanove". In: *Proceedings of the 2012 ACM Conference on Computer Supported Cooperative Work*, p. 1175.
- Metta, G., G. Sandini, D. Vernon, L. Natale, and F. Nori (2008). "The iCub humanoid robot: an open platform for research in embodied cognition." In: *Proceedings of the 8<sup>th</sup> Workshop on Performance Metrics for Intelligent Systems*. New York, New York, USA: ACM Press, p. 50.

- Michalowski, M., K. Machulis, M. Gasson, and T. Hersan (2013). *Hack "My Keepon" With an Arduino Brain*. URL: <http://makezine.com/projects/make-35/my-franken-keepon/>.
- Millner, A. and E. Baafi (2011). "Modkit: blending and extending approachable platforms for creating computer programs and interactive objects". In: *Proceedings of the 10<sup>th</sup> International Conference on Interaction Design and Children*, pp. 250–253.
- Milto, E., C. Rogers, and M. Portsmore (2002). "Gender differences in confidence levels, group interactions, and feelings about competition in an introductory robotics course". In: *Proceedings of the 32<sup>nd</sup> Annual Frontiers in Education Conference*. Vol. 2. IEEE, F4C–7–F4C–14.
- Mitchell, G. (2012). "The Raspberry Pi single-board computer will revolutionise computer science teaching". In: *Engineering & Technology* 7, p. 26.
- Modi, K., J. Schoenberg, and K. Salmond (2012). *Generation STEM: What Girls Say about Science, Technology, Engineering, and Math*. Tech. rep., pp. 1–44.
- Mondada, F., M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klapotocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli (2009). "The e-puck, a Robot Designed for Education in Engineering". In: *Proceedings of the 9<sup>th</sup> Conference on Autonomous Robot Systems and Competitions*. Vol. 1. 1, pp. 59–65.
- Mori, M. (1970). "The Uncanny Valley". In: *Energy* 7.4, pp. 33–35.
- Mostert-Van Der Sar, M., I. Mulder, L. Remijn, and P. Troxler (2013). "FabLabs in design education". In: *Proceedings of the 15<sup>th</sup> International Conference on Engineering and Product Design Education*. September, pp. 629–634.
- Mueller, S. and P. Baudisch (2015). "Laser Cutters: A New Class of 2D Output Devices". In: *ACM Interactions* 22.5, pp. 72–74.
- Nakamura, J. and M. Csikszentmihalyi (2002). "The Concept of Flow". In: *The Handbook of Positive Psychology*. Oxford University Press, pp. 89–92.
- Naylor, G. (1980). *The Arts and Crafts Movement: A Study of Its Sources, Ideals, and Influence on Design Theory*. MIT Press, p. 208.
- Nimer, J. and B. Lundahl (2007). "Animal-assisted therapy: A meta-analysis". In: *Anthrozoos* 20.3, pp. 225–238.
- Nishio, S., H. Ishiguro, and N. Hagita (2007). "Geminoid: Teleoperated android of an existing person". In: *Humanoid Robots: New Developments*. Ed. by A. C. d. P. Filho. June. I-Tech Education and Publishing, pp. 343–352.
- Nof, S. Y., ed. (1999). *Handbook of Industrial Robots*. Second edi. John Wiley & Sons.
- Norman, D. A. (2002). "Emotion & design: attractive things work better". In: *Interactions* 9.4, pp. 36–42.
- (2009). "Systems Thinking: A Product Is More Than the Product". In: *Interactions* 16.5, pp. 52–54.
- (2013). *The Design of Everyday Things*. Revised an. Vol. 16. 4. Basic Books, p. 272.
- Norton, M., D. Mochon, and D. Ariely (2012). "The IKEA effect: When labor leads to love". In: *Journal of Consumer Psychology* 22.3, pp. 453–460.
- Novikova, J. and L. Watts (2015). "Towards Artificial Emotions to Assist Social Coordination in HRI". In: *International Journal of Social Robotics* 7.1, pp. 77–88.
- Ogawa, K., S. Nishio, T. Minato, and H. Ishiguro (2012). "Android Robots as Telepresence Media". In: *Biomedical Engineering and Cognitive Neuroscience for Healthcare*:

- Interdisciplinary Applications*. Ed. by J. Wu. Medical Information Science Reference, pp. 54–63.
- Oh, H. and M. D. Gross (2015). “Cube-in: A Learning Kit for Physical Computing Basics”. In: *Proceedings of the 9<sup>th</sup> International Conference on Tangible, Embedded, and Embodied Interaction*. Stanford, pp. 383–386.
- Osborne, R. B., A. J. Thomas, and J. R. N. Forbes (2010). “Teaching with robots: a service-learning approach to mentor training”. In: *Proceedings of the 41<sup>st</sup> ACM Technical Symposium on Computer Science Education*, pp. 172–176.
- Oung, R. and R. D’Andrea (2011). “The Distributed Flight Array”. In: *Mechatronics* 21.6, pp. 908–917.
- Pacheco, M., R. Fogh, H. H. Lund, and D. J. Christensen (2015). “Fable II: Design of a modular robot for creative learning”. In: *Proceedings of the 2015 IEEE International Conference on Robotics and Automation*, pp. 6134–6139.
- Pack, D. and R. Avanzato (2004). “Fire-fighting mobile robotics and interdisciplinary design-comparative perspectives”. In: *IEEE Transactions on Education* 47.3, pp. 369–376.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.
- (1986). *Constructionism: A New Opportunity for Elementary Science Education*. Tech. rep. Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group.
- Papert, S. and I. Harel (1991). “Situating Constructionism”. In: *Constructionism*. Ablex Publishing, pp. 1–12.
- Paradiso, J. A., J. Heidemann, and T. G. Zimmerman (2008). “Hacking Is Pervasive”. In: *IEEE Pervasive Computing* 7.3, pp. 13–15.
- Paulos, E., R. Honicky, and B. Hooker (2009). “Citizen Science: Enabling Participatory Urbanism”. In: *Handbook of Research on Urban Informatics: The Practice and Promise of the Real-Time City*, pp. 414–436.
- Pearce, J. M. (2012). “Building Research Equipment with Free, Open-Source Hardware”. In: *Science* 337.6100, pp. 1303–1304.
- Pearce, J. M. (2013). *Open-Source Lab: How to Build Your Own Hardware and Reduce Research Costs*. Elsevier, p. 240.
- Pearson, G. and A. T. Young, eds. (2002). *Technically Speaking: Why All Americans Need to Know More About Technology*. National Academies Press, p. 170.
- Peterson, T. F. (2011). *Nightwork: A History of Hacks and Pranks at MIT*, p. 232.
- Pettis, B. (2012). *Let’s try that again*. URL: <http://www.makerbot.com/media-center/2012/09/24/lets-try-that-again>.
- Phillips, R., S. Baurley, and S. Silve (2014). “Citizen Science and Open Design: Workshop Findings”. In: *Design Issues* 30.4, pp. 52–66.
- Piro, J. (2010). “Going From STEM to STEAM”. In: *Education Week* 29.24, pp. 28–29.
- Plutchik, R. (1980). “A general psychoevolutionary theory of emotion”. In: *Emotion: Theory, Research, and Experience* 1.3, pp. 3–33.
- Pop, C. A., R. Simut, S. Pintea, J. Saldien, A. Rusu, D. David, J. Vanderfaecillie, D. Lefebber, and B. Vanderborcht (2013). “Can the Social Robot Probo Help Children With Autism To Identify Situation-Based Emotions? a Series of Single Case Experiments”. In: *International Journal of Humanoid Robotics* 10.3, p. 1350025.

- Powell, A. (2012). "Democratizing production through open source knowledge: from open software to open hardware". In: *Media, Culture & Society* 34.6, pp. 691–708.
- Prince, M. (2004). "Does Active Learning Work? A Review of the Research". In: *Journal of Engineering Education* 93, July, pp. 223–231.
- Pugh, S. (1991). *Total design: integrated methods for successful product engineering*. Wokingham: Addison-Wesley.
- Quigley, M., A. Asbeck, and A. Ng (2011). "A low-cost compliant 7-DOF robotic manipulator". In: *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, pp. 6051–6058.
- Quigley, M., K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg (2009). "ROS: an open-source Robot Operating System". In: *Proceedings of the 2009 International Conference on Robotics and Automation - Open-source software workshop*. Kobe, Japan.
- Ranganathan, P., R. Schultz, and M. Mardani (2008). "Use of LEGO NXT Mindstorms brick in engineering education". In: *Proceedings of the 2008 ASEE North Midwest Sectional Conference*, pp. 17–19.
- Raymond, E. (1999). "The cathedral and the bazaar". In: *Knowledge, Technology & Policy* 12.3, pp. 23–49.
- Reas, C. and B. Fry (2007). *Processing: A Programming Handbook for Visual Designers and Artists*. The MIT Press, p. 736.
- Resnick, M., J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. a. Y. Silver, B. Silverman, and Y. Kafai (2009). "Scratch: Programming for All." In: *Communications of the ACM* 52, pp. 60–67.
- Riedo, F., M. Chevalier, S. Magnenat, and F. Mondada (2013). "Thymio II, a robot that grows wiser with children". In: *Proceedings of the 2013 IEEE Workshop on Advanced Robotics and its Social Impacts*, pp. 187–193.
- Riedo, F., P. Rétornaz, L. Bergeron, N. Nyffeler, and F. Mondada (2012). "A two years informal learning experience using the Thymio robot". In: *Advances in Autonomous Mini Robots*. Springer, pp. 37–48.
- Riojas, M., S. Lysecky, and J. Rozenblit (2012). "Educational Technologies for Precollege Engineering Education". In: *IEEE Transactions on Learning Technologies* 5.1.
- Robins, B., K. Dautenhahn, R. T. Boekhorst, and A. Billard (2005). "Robotic assistants in therapy and education of children with autism: Can a small humanoid robot help encourage social interaction skills?" In: *Universal Access in the Information Society* 4.2, pp. 105–120.
- Rockland, R., D. Bloom, and J. Carpinelli (2010). "Advancing the "E" in K-12 STEM education". In: *Journal of Technology Studies* 36.1, pp. 53–65.
- Rogers, S. J. (1996). "Brief report: early intervention in autism." In: *Journal of autism and developmental disorders* 26.2, pp. 243–246.
- Rosner, D. K. (2009). "Learning from IKEA Hacking: "I'm Not One to Decoupage a Tabletop and Call It a Day". In: *Proceedings of the 2009 SIGCHI Conference on Human Factors in Computing Systems*, pp. 419–422.
- Royce, W. W. (1970). "Managing the Development of Large Software Systems". In: *Proceedings of IEEE WESCON*, pp. 328–338.
- Russel, J. (1980). "A circumplex model of affect". In: *Journal of Personality and Social Psychology*.

- Saldien, J. (2009). "The Development of the Huggable Social Robot Probo. On the Conceptual Design and the Software Architecture." PhD thesis. Vrije Universiteit Brussel, p. 163.
- Saldien, J., K. Goris, B. Vanderborght, J. Vanderfaellie, and D. Lefeber (2010). "Expressing emotions with the social robot Probo". In: *International Journal of Social Robotics* 2.4, pp. 377–389.
- Saleiro, M., B. Carmo, J. M. F. Rodrigues, and J. M. H. Du Buf (2013). "A low-cost classroom-oriented educational robotics system". In: *Lecture Notes in Computer Science*. Vol. 8239 LNAI, pp. 74–83.
- Sanders, E. B.-N. (2006). "Design Serving People". In: *Cumulus Working Papers* 15.5, pp. 28–33.
- Sanders, E. B.-N. and P. J. Stappers (2008). "Co-creation and the new landscapes of design". In: *CoDesign* 4.1, pp. 5–18.
- Sato, M., I. Poupyrev, and C. Harrison (2012). "Touché: enhancing touch interaction on humans, screens, liquids, and everyday objects". In: *Proceedings of the 2012 SIGCHI Conference on Human Factors in Computing Systems*. c, pp. 483–492.
- Schelly, C., G. Anzalone, B. Wijnen, and J. M. Pearce (2015). "Open-source 3-D printing technologies for education: Bringing additive manufacturing to the classroom". In: *Journal of Visual Languages & Computing* 28, pp. 226–237.
- Schmidt, E. and J. Cohen (2013). *The New Digital Age: Transforming Nations, Businesses, and Our Lives*. Knopf Doubleday Publishing Group.
- Schreiner, C. and S. Sjøberg (2010). *The ROSE project An overview and key findings*. Tech. rep. University of Oslo, pp. 1–31.
- Schweikardt, E. (2011). "Modular robotics studio". In: *Proceedings of the 5<sup>th</sup> International Conference on Tangible, Embedded, and Embodied Interaction*. New York, New York, USA: ACM Press, p. 353.
- Silver, J. S. (2014). "Block x Lens - World as Construction Kit". PhD thesis. MIT.
- Silver, J., E. Rosenbaum, and D. Shaw (2012). "Makey Makey: Improvising Tangible and Nature-Based User Interfaces". In: *Proceedings of the 6<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*. Ontario: ACM, p. 367.
- Simon, T. M., B. H. Thomas, R. T. Smith, and M. Smith (2014). "Adding input controls and sensors to RFID tags to support dynamic tangible user interfaces". In: *Proceedings of the 8<sup>th</sup> International Conference on Tangible Embedded and Embodied Interaction*, pp. 165–172.
- Sproewitz, A., L. Kuechler, A. Tuleu, M. Ajallooeian, M. D'Haene, R. Moeckel, and A. J. Ijspeert (2011). "Oncilla Robot, A Light-weight Bio-inspired Quadruped Robot for Fast Locomotion in Rough Terrain". In: *Proceedings of the 2011 Symposium on Adaptive Motion of Animals and Machines*, pp. 63–64.
- Stajano, F. (2000). "Python in education: Raising a generation of native speakers". In: *Proceedings of the 8<sup>th</sup> International Python Conference*.
- Stiehl, W. D., J. K. Lee, C. Breazeal, M. Nalin, A. Morandi, and A. Sanna (2009). "The Huggable: A Platform for Research in Robotic Companions for Pediatric Care". In: *Proceedings of the 8<sup>th</sup> International Conference on Interaction Design and Children*, pp. 317–320.

- Stoelen, M., F. Bonsignorio, and A. Cangelosi (2016). "Co-exploring actuator antagonism and bio-inspired control in a printable robot arm". In: *From Animals to Animats 14*. Ed. by E. Tuci, A. Giagkos, M. Wilson, and J. Hallam. Springer, pp. 244–255.
- Sundar, S. S., T. F. Waddell, and E. H. Jung (2016). "The Hollywood robot syndrome: Media effects on older adults' attitudes toward robots and adoption intentions". In: *Proceedings of the 11<sup>th</sup> ACM/IEEE International Conference on Human-Robot Interaction*, pp. 343–350.
- Takayama, L., W. Ju, and C. Nass (2008). "Beyond dirty, dangerous and dull: what everyday people think robots should do". In: *Proceedings of the 3<sup>rd</sup> International Conference on Human-Robot Interaction*. Amsterdam, pp. 25–32.
- TAPR (2007). *The TAPR Open Hardware License Version 1.0 (May 25, 2007)*. URL: [https://www.tapr.org/TAPR\\_Open\\_Hardware\\_License\\_v1.0.txt](https://www.tapr.org/TAPR_Open_Hardware_License_v1.0.txt).
- Tenzer, Y., L. P. Jentoft, and R. D. Howe (2014). "The feel of MEMS barometers: Inexpensive and easily customized tactile array sensors". In: *IEEE Robotics and Automation Magazine* 21.3, pp. 89–95.
- Terryn, R., S. Flamand, J. Saldien, P. Deconinck, F. wyffels, and S. Verstockt (2016). "Innovative Design of a Hexapod Scorpion Through Digital Production Techniques". In: *Proceedings of the 19<sup>th</sup> International Conference on Climbing and Walking Robots*. London: World Scientific Publishing Company, pp. 1–8.
- Thomaszewski, B., S. Coros, D. Gauge, V. Megaro, E. Grinspun, and M. Gross (2014). "Computational Design of Linkage-based Characters". In: *ACM Transactions on Graphics* 33.4, 64:1–64:9.
- Thompson, C. (2008). "Build It. Share It. Profit. Can Open Source Hardware Work?" In: *WIRED*.
- Tiger, L. (1992). *The Pursuit of Pleasure*. Transaction Publishers, p. 330.
- Tilden, M. (2013). *Mark Tilden on "What is the single biggest obstacle preventing robotics from going mainstream?"* URL: <http://robohub.org/mark-tilden-on-what-is-the-single-biggest-obstacle-preventing-robotics-from-going-mainstream/> (visited on 10/02/2015).
- Törnkvist, S. (1998). "Creativity: can it be taught? The case of Engineering Education". In: *European Journal of Engineering Education* 23.1, pp. 5–12.
- Troxler, P. (2013). "Making the Third Industrial Revolution. The Struggle for Polycentric Structures and New Peer-Production Commons in the FabLab Community." In: *FabLab – Of Machines, Makers and Inventors*. Ed. by J. Walter-Herrmann and C. Büching, pp. 181–195.
- Tryggvason, G. and D. Apelian (2006). "Re-engineering engineering education for the challenges of the 21st century." In: *Journal of the Minerals, Metals and Materials Society* 58.10, pp. 14–17.
- Tsagarakis, N. G., G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, A. J. Ijspeert, M. C. Carrozza, and D. G. Caldwell (2007). "iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research". In: *Journal of Advanced Robotics* 21.10, pp. 1151–1175.
- Tseng, T. and M. Resnick (2014). "Product versus process". In: *Proceedings of the 2014 Conference on Designing Interactive Systems*. New York, New York, USA: ACM Press, pp. 425–428.



- Van den Broeck, M. (2016). “Het ontwerpen van nieuwe functionaliteiten voor een sociale robot in de context van de zorg en therapie”. Master’s thesis. Ghent University, p. 162.
- Vanderborght, B., R. Simut, J. Saldien, C. Pop, A. S. Rusu, S. Pinte, D. Lefebvre, and D. O. David (2012). “Using the social robot Probo as a social story telling agent for children with ASD”. In: *Interaction Studies* 13.3, pp. 348–372.
- Vandeveld, C., P. Conradie, J. De Ville, and J. Saldien (2014). “Playful interaction: designing and evaluating a tangible rhythmic musical interface”. In: *Proceedings of INTER-FACE: The Second International Conference on Live Interfaces*. Ed. by A. Sa, M. Carvalhais, and A. McLean. Lisbon, Portugal, pp. 132–141.
- Vandeveld, C. and J. Saldien (2016a). “An Open Platform for the Design of Social Robot Embodiments for Face-to-Face Communication”. In: *Proceedings of the 11<sup>th</sup> International Conference on Human-Robot Interaction*. Christchurch, New Zealand, pp. 287–294.
- (2016b). “Demonstration of OPSORO – an Open Platform for Social Robots”. In: *Proceedings of the 11<sup>th</sup> ACM/IEEE International Conference on Human-Robot Interaction*. Christchurch, New Zealand.
- Vandeveld, C., J. Saldien, C. Ciocci, and B. Vanderborght (2013a). “Overview of technologies for building robots in the classroom”. In: *Proceedings of the 4<sup>th</sup> International Conference on Robotics in Education*.
- (2013b). “Systems overview of Ono: a DIY reproducible open source social robot”. In: *Lecture Notes in Computer Science*. Ed. by G. Herrmann, M. J. Pearson, A. Lenz, P. Bremner, A. Spiers, and L. U. Vol. 8239. Springer International Publishing, pp. 311–320.
- (2013c). “The use of social robot ono in robot assisted therapy”. In: *International Conference on Social Robotics, Proceedings*.
- (2014). “Ono, a DIY open source platform for social robotics”. In: *Proceedings of the 8<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*. Ed. by A. Butz, S. Greenberg, S. Bakker, L. Loke, and A. De Luca.
- Vandeveld, C., M. Vanhoucke, and J. Saldien (2015). “Prototyping social interactions with DIY animatronic creatures”. In: *Proceedings of the 9<sup>th</sup> International Conference on Tangible, Embedded and Embodied Interaction*.
- Vandeveld, C., F. Wyffels, C. Ciocci, B. Vanderborght, and J. Saldien (2015). “Design and evaluation of a DIY construction system for educational robot kits”. In: *International Journal of Technology and Design Education* 26.4, pp. 521–540.
- Vandeveld, C., F. Wyffels, B. Vanderborght, and J. Saldien (in press). “DIY Design for Social Robots”. In: *IEEE Robotics and Automation Magazine*.
- Vanhoucke, M., C. Vandeveld, J. Ingels, G. Hamon, and J. Saldien (2014). “Serious game as educational tool for safety and prevention”. In: *Proceedings of the 2014 ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play – Workshop Participatory Design for Serious Game Design*. Toronto, Canada: ACM.
- Vastenburg, M., N. Romero Herrera, D. Van Bel, and P. Desmet (2011). “PMRI”. In: *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems - Extended Abstracts*. New York, New York, USA: ACM Press, p. 2155.
- VDAB (2012). *VDAB ontcijfert nr 28*. Tech. rep. VDAB Brussels.
- Veretennicoff, I., J. Vandewalle, B. Seghers, C. Aerts, Y. Bruynseraede, P. Cara, W. Dehaene, B. Hendrickx, C. Hirsch, R. Hostyn, C. Malcorps, N. Schamp, A. Sevrin, K.

- Strubbe, D. V. Dyck, P. V. Houtte, V. V. Speybroeck, and J. Willems (2015). *De STEM-leerkracht*. 38, p. 56.
- Verner, I. M. and D. J. Ahlgren (2004). "Robot contest as a laboratory for experiential engineering education". In: *Journal on Educational Resources in Computing* 4.2.
- Von Hippel, E. and J. A. Paradiso (2008). "User innovation and hacking". In: *IEEE Pervasive Computing* 7.3, pp. 66–69.
- Vygotsky, L. S. (1978). "Development of Higher Psychological Processes". In: *Mind in Society*. Ed. by M. Cole, J. V. Steiner, S. Scribner, and E. Souberman. Harvard University Press.
- Wainer, J., D. J. Feil-Seifer, D. a. Shell, and M. J. Matarić (2006). "The role of physical embodiment in human-robot interaction". In: *Proceedings of the 2006 IEEE International Workshop on Robot and Human Interactive Communication*, pp. 117–122.
- Walter-Herrmann, J. and C. Büching (2013). *FabLab: Of Machines, Makers and Inventors*. Bielefeld: Transcript Verlag.
- Wang, Z.-Y., X.-L. Ding, and A. Rovetta (2010). "Analysis of typical locomotion of a symmetric hexapod robot". In: *Robotica* 28.6, pp. 893–907.
- Weiser, M. (1994). "The world is not a desktop". In: *Interactions* 1.1, pp. 7–8.
- Westlund, J. K., J. J. Lee, L. Plummer, F. Faridi, J. Gray, M. Berlin, H. Quintus-bosz, R. Hartmann, M. Hess, S. Dyer, and K. Santos (2016). "Tega: A Social Robot". In: *Proceedings of the 11<sup>th</sup> International Conference on Human-Robot Interaction*, p. 8370.
- Williams, A., A. Gibb, and D. Weekly (2012). "Research with a Hacker Ethos: What DIY Means for Tangible Interaction Research". In: *Interactions* 19.2, pp. 14–19.
- Williams, K., I. Igel, R. Poveda, V. Kapila, and M. Iskander (2012). "Enriching K-12 science and mathematics education using LEGOs". In: *Advances in Engineering Education* 3.2.
- Williams, K. and C. Breazeal (2013). "Reducing driver task load and promoting sociability through an Affective Intelligent Driving Agent (AIDA)". In: *Lecture Notes in Computer Science*. Ed. by P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, and M. Winckler. Vol. 8120, pp. 619–626.
- Wilson, D. M. (1966). "Insect Walking". In: *Annual Review of Entomology* 11, pp. 103–122.
- Windisch, K. G. von (1783). *Briefe über den Schachspieler des Hrn. von Kempelen, nebst drei Kupferstichen die diese berühmte Maschine vorstellen*.
- Wolf, M. and S. McQuitty (2011). "Understanding the do-it-yourself consumer: DIY motivations and outcomes". In: *Academy of Marketing Science Review* 1, pp. 154–170.
- wyffels, F., K. Bruneel, P. Bertels, M. D'Haene, W. Heirman, and T. Waegeman (2012). "A human-friendly way of programming robots". In: *Proceedings of the 5<sup>th</sup> International Workshop on Human-Friendly Robotics*.
- wyffels, F., M. Hermans, and B. Schrauwen (2010). "Building robots as a tool to motivate students into an engineering education". In: *AT&P Journal Plus* 2010.2, pp. 113–116.
- Yanamandram, V. M. K. and J. H. Panchal (2014). "Evaluating the Level of Openness in Open Source Hardware". In: *Product Development in the Socio-sphere: Game Changing Paradigms for 21<sup>st</sup> Century Breakthrough Product Development and Innovation*, pp. 99–120.

- Yanco, H. A. and J. L. Drury (2004). "Classifying human-robot interaction: an updated taxonomy." In: *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*.
- Zappi, V. and A. McPherson (2014). "Design and Use of a Hackable Digital Instrument". In: *Proceedings of the 2<sup>nd</sup> International Conference on Live Interfaces*. Lisbon, pp. 208–219.
- Zimmermann, L. (2014). "Business". In: *Building Open Source Hardware: DIY Manufacturing for Hackers and Makers*. Addison-Wesley Professional.
- Zubrycki, I. and G. Granosik (2016). "Understanding Therapists' Needs and Attitudes Towards Robotic Support. The Robotrapia Project". In: *International Journal of Social Robotics*.



# LIST OF TABLES

- 1.1 Summary of the design goals ..... 68
- 2.1 System Usability Scale questionnaire. .... 84
- 2.2 Excerpt from the AttrakDiff questionnaire. .... 85
- 2.3 Summary of UX measurement tools ..... 88
- 2.4 Examples of digital fabrication techniques ..... 92
- 2.5 Examples of using design complexity of digital manufacturing techniques \_ 94
  
- 3.1 Comparison of the three building systems ..... 118
- 3.2 Q2 – Robot ratings ..... 122
- 3.3 Q3 – Systems ranking by the experts ..... 123
- 3.4 Summary of concept design requirements ..... 129
- 3.5 Cost breakdown of the v1.0 prototype ..... 133
- 3.6 Interaction rates ..... 139
- 3.7 Emotion recognition rates ..... 139
- 3.8 Results of the questionnaire’s open questions ..... 159
- 3.9 Overview of robot concepts, module use, and skinning techniques ..... 166
- 3.10 Overview of the iterative design process ..... 176
  
- 5.1 Overview of current users ..... 236



# LIST OF FIGURES

1.1	Arduino Uno	28
1.2	RepRap Prusa Mendel i3	28
1.3	Model of long tail economics.	35
1.4	LEGO Mindstorms EV3 kit	43
1.5	Thymio. Adopted from Riedo, Chevalier, et al. (2013).	43
1.6	Examples of programmable controllers.	45
1.7	Text-based programming languages.	46
1.8	Visual programming language made with Blockly.	46
1.9	Examples of construction systems.	48
1.10	The consumer/designer spectrum.	51
1.11	Csikszentmihalyi's model of flow.	51
1.12	A designer's perspective of flow.	53
1.13	Model of da Vinci's mechanical knight <sup>3</sup> .	54
1.14	The Mechanical Turk.	54
1.15	Examples of commercial IoT products.	55
1.16	Three types of robots	58
1.17	Baxter.	58
1.18	Kismet.	58
1.19	Travis.	61
1.20	Nao.	61
1.21	iCub.	63
1.22	Hacked MyKeepon	63
1.23	Poppy Humanoid.	64
1.24	InMoov.	64
1.25	Redesigned Probo head.	65
1.26	Moodles.	65
2.1	Lifecycle model of conventional products.	73

2.2	Lifecycle model of platform-based products. _____	74
2.3	ISO 9241-210 model for human-centered design of interactive systems. ____	76
2.4	Waterfall model of development. _____	76
2.5	Four patterns of reasoning. _____	78
2.6	Hierarchy of human needs vs. a hierarchy of consumer needs. _____	81
2.7	The four pleasures. _____	81
2.8	Hassenzahl's model of user experience. _____	83
2.9	Pick-A-Mood expressions of eight mood types and neutral. _____	86
2.10	Example of a user experience curve. _____	86
2.11	Example illustrating Pugh's model of controlled convergence. _____	89
2.12	Using asymmetry to improve the assembly process. _____	95
3.1	Schematic overview of the design iterations. _____	98
3.2	Final prototype of the Stigmergic Ant _____	99
3.3	Top view showing tip position at $T$ , body coordinate system at $O$ , and leg coordinate system at $A$ _____	102
3.4	Kinematic chain of the leg _____	103
3.5	Tripods of the alternating tripod gait _____	104
3.6	Effect of the displacement vectors on leg tip positions _____	104
3.7	Final prototype of the scorpion _____	106
3.8	Playfields used during the contest. _____	110
3.9	The robot kit used in the contest. _____	111
3.10	Design funnel – schematic overview of our design process. _____	113
3.11	Student designs from iteration one. _____	113
3.12	Prototypes from iteration two. _____	114
3.13	Laser-cut screw connector system. _____	115
3.14	Printed friction-fit connector system. _____	116
3.15	Printed "hybrid" connector system. _____	117
3.16	PMRI results. _____	121
3.17	AttrakDiff results. _____	121
3.18	Q1 – Experts' criteria for grading robots. _____	122
3.19	Children interacting with Ono v1.1 through the control box _____	125
3.20	Changes in the slot-and-tab construction system between v1.1 (left) and v1.2 (right) _____	126
3.21	Exploration sketches for the embodiment design _____	127
3.22	The Uncanny Valley. _____	128
3.23	Rendered image of the Ono outer skin surface model _____	129
3.24	The Ono v1.0 prototype with control unit _____	130
3.25	CAD model showing construction and DOFs of Ono. The servo actuators are shown in a darker color. _____	132



3.26	Assembly instructions for the Ono v1.0 eye module	133
3.27	Line drawing of the assembled frame	134
3.28	Participant assembling the eye module.	134
3.29	Child interacting with Ono during the pilot study	137
3.30	Wizard of Oz setup to test the child controller concept.	140
3.31	Comparison of the first generation module interface (left) vs. second generation module interface (right)	144
3.32	Comparison of the first generation frame (left) vs. second generation frame (right) construction	144
3.33	Mechanism of the eye module. Old direct-drive mechanism (left) vs. new linkage-driven mechanism (right).	145
3.34	Participants of the workshop posing around the robot they built in one day	146
3.35	Workshop participants assembling an Ono.	148
3.36	Results of the AttrakDiff questionnaire.	150
3.37	Results from the Likert scale questions.	151
3.38	UX curves drawn by the the participants of the workshop.	151
3.39	Quiz control interface	153
3.40	Blockly API issues	153
3.41	Interface mockup	154
3.42	Social Script app in use	154
3.43	Participants working on the “Michael Jackson” robot.	156
3.44	UX curves drawn by the the participants of the workshop.	160
3.45	Results of the AttrakDiff questionnaire.	160
3.46	Design process overview of one robot.	164
3.47	The ten robots designed during the student course.	165
3.48	Results from the Likert scale questions.	168
3.49	Relative difficulty of each phase.	169
3.50	Participants building custom robots using the Opsoro modules and craft materials.	171
3.51	Grid connection system used in the Frankfurt experiment	172
3.52	Cardboard embodiments designed by the participants.	173
3.53	Results of the AttrakDiff-Short questionnaire.	174
3.54	Results from the Likert scale questions.	174
4.1	CAD model of the mouth module	179
4.2	Mouth module prototype	179
4.3	Exploded view of the module	180
4.4	CAD model of the eyebrow module	181
4.5	Eyebrow module prototype	181

4.6	Eye module prototype	181
4.7	Ball joint linkages	182
4.8	Eyeball pivot mechanism	182
4.9	Exploded view of the module	183
4.10	Connection between servo spline and laser-cut gear.	184
4.11	CAD model of the joint module	184
4.12	Joint module prototype	184
4.13	Exploded view of the module	185
4.14	CAD model of 1 <sup>st</sup> neck module	186
4.15	Prototype of 1 <sup>st</sup> neck module	186
4.16	Prototype of 2 <sup>nd</sup> neck module	187
4.17	1 <sup>st</sup> workshop base	188
4.18	2 <sup>nd</sup> workshop base	188
4.19	Example of the snap connectors used in the frame design	190
4.20	Architecture of the frame of Ono, showing four sub-units	191
4.21	Developable surface (left) vs. doubly curved surface (right)	192
4.22	Foam flattening process.	193
4.23	Laser-cut snap sockets.	195
4.24	Picture of the Arduino shield	196
4.25	Picture of HAT rev. 0	198
4.26	Picture of HAT rev. 1	199
4.27	System diagram of HAT rev. 1	199
4.28	Picture of HAT rev. 2	201
4.29	System diagram of HAT rev. 2	201
4.30	SPI write protocol	202
4.31	SPI read protocol	202
4.32	Capacitive touch sensor data.	205
4.33	Servo PWM timing	206
4.34	Circumplex model of affect.	208
4.35	Two-step interpolation of the circumplex model	209
4.36	Example of the algorithm applied to a 2-DOF eyebrow module	211
4.37	Mapping DOF positions to pulse widths	211
4.38	Interpolation using the NumPy module	214
4.39	Hardware interface	215
4.40	Web interface	215
4.41	Main page of the Opsoro interface, showing all apps	216
4.42	Example showing the interface of the “Social Script” app	216
4.43	High-level overview of the Opsoro software architecture	218
4.44	Authentication scheme	219

4.45	App manager – setup	221
4.46	App manager – activation	221
4.47	App manager – interaction	221
4.48	Circumplex Interface App	223
4.49	Sounds App	223
4.50	Config Editor App	224
4.51	Touch Graph App	224
4.52	Sliders App	225
4.53	Lua Scripting App	225
4.54	Visual Programming App	226
4.55	Social Script App	227
5.1	Map of current users of the social robot Ono and the Opsoro Starter Kit.	235
5.2	OTO – a mobile extension for the Opsoro platform	236
5.3	Opsoro components produced by a farm of eight low-cost 3D printers.	240