Modular Robotic Playware

Pacheco, Moises

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

Technical University of Denmark

Moises Pacheco

Modular Robotic Playware

PhD Thesis, May 2016

DTU Electrical Engineering
Department of Electrical Engineering

Modular Robotic Playware

Moises Pacheco

Technical University of Denmark
Kgs. Lyngby, Denmark, 2016

Technical University of Denmark
Automation and Control (AUT)
Elektrovej Building 326
DK-2800, Kgs. Lyngby
Denmark
Phone: (+45) 45 25 35 76
Email: info@elektro.dtu.dk
www.elektro.dtu.dk

ISBN: 978-87-92465-61-0

Summary

There is an open debate on whether today's consumer driven economy is constantly encouraging us to buy and not to build. Plagiarism and copyright policies can potentially serve as mental barricades that dry out our curiosity, creativity and collaboration [1, 2]. In this work we, at Center for Playware, seek to revitalize and quench our users thirst for knowledge within the domain of robotics. We believe that, given the right tools, anyone can become a robot designer. As a result we have developed Fable a modular robot designed for education that can help motivate users towards social interaction while building and playing with robots. Our approach uses easy to assemble units that allow users to build a robot in a matter of seconds. Fable's design encloses topics in modular robotics within the Danish educational sector.

Fable is a heterogeneous chain based user-reconfigurable robot, that is based on two types of modules: passive and active. Passive modules give the system shape and structure, while active modules are able to interact with the environment. Fable's library of modules currently consists of a 2 DOF Joint Module, a Wheel Module a Sensor Module and a set of 11 different Passive Modules. The system also has a Dongle, used to enable PCs to communicate wirelessly with active modules. Users can choose the communication channel by pressing the button on the Dongle, where each radio channel is color coded. Fable uses scaffolding based progression by allowing unexperienced users to program through a graphical user interface based on Google's "Blockly" and later progress to Python programming. The system was tested with more than 500 students in various settings including: public schools, after-school clubs and research labs. We then used this knowledge to improve our design and built a robot that fits the requirements of the Danish educational system. In autumn 2016 Fable will be available to public schools in Denmark.

Resumé

Verden i dag er fyldt med forbrugerprodukter, der konstant tilskynder os til at købe og forbruge, ikke til at bygge og skabe. Fra en tidlig alder fungerer plagiat- og ophavsretspolitikker som mental barrikader, der udtørrer vores nysgerrighed, kreativitet og samarbejde.

I dette arbejde søger vi, på Center for Playware, at genoplive vores brugeres tørst efter viden inden for området robotteknologi. Vi mener at enhver, givet de rette værktøjer, kan blive en robotdesigner. Derfor har vi udviklet Fable, en modulær robot, designet til uddannelsessektoren, der motiverer brugerne til social interaktion, samtidig med at de bygger og leger med robotter. Vores tilgang er anvendelsen af enheder, der er lette at samle, og som giver brugerne mulighed for at bygge en robot i løbet af få sekunder. Fable's design dækker den danske uddannelsessektors fagområder inden for modulære robotter.

Fable er en heterogen kæde-baseret bruger- rekonfigurerbar robot, der er baseret på to typer moduler: passive og aktive. Passive moduler givet systemet form og struktur, medens de aktive moduler er i stand til at interagere med miljøet de opererer i. Fable's bibliotek af moduler består af 2 DOF led-modul, et hjul-modul og et sensor-modul samt et sæt af 11 forskellige passive moduler. Systemet har også en dongle, der muliggør trådløs kommunikation imellem en PC og de aktive moduler. Brugerne kan vælge kommunikationskanal ved at trykke på donglens knap, hvorved hver radiokanal er farvekodet. Fable anvender Scaffolding baseret progression, da den giver den uerfarende bruger muligheden for at programmere via en grafisk brugergrænseflade, baseret på Google's "Blockly", for senere at arbejde videre over i Python programmering. Systemet er afprøvet på mere end 500 elever i forskellige test situationer: folkeskolen, skolefritidsordninger og forskningslaboratorier. Vi har brugt vores viden herfra, til at forbedre vores design og bygge en robot, der passer til kravene i det danske uddannelsessystem. I efteråret 2016 vil Fable være tilgængelig for uddannelsesinstitutioner i Danmark.

Preface

This thesis is written as a conclusion of my PhD project at the Technical University of Denmark, at the Center for Playware section of the Department of Electrical Engineering. My Ph.D. project was partially funded by the Danish National Advanced Technology Foundation

The research that I present in this document was carried out between December 2012 and April 2016 and supervised by professor Henrik Hautop Lund and associate professor David Johan Christensen. I based the structure of the thesis on a collection of peer reviewed scientific articles, that I wrote throughout the 3 year Ph.D. project.

The articles are presented in the appendices. The main focus of my work has been in the design and development of modular robots for everyday use in education, research and leisure.

Moises Pacheco
Kgs. Lyngby 2015

List of Publications

The following papers are included in the thesis and are listed in chronological order.

(A) M. Pacheco, M. Moghadam, A. Magnússon, B. Silverman, H. H. Lund, and D. J. Christensen. "Fable: Design of a modular robotic playware platform". *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013.

(B) A. Magnússon, M. Pacheco, M. Moghadam, H. H. Lund, and D. J. Christensen. "Fable: Socially interactive modular robot". *The Eighteenth International Symposium on Artificial Life and Robotics 2013*, 2013.

(C) B. Heesche, E. MacDonald, R. Fogh, M. Pacheco, and D. J. Christensen. "Playful interaction with voice sensing modular robots". *Social Robotics*, 2013.

(D) M. Pacheco, R. Fogh, H. H. Lund, and D. J. Christensen. "Fable: A Modular Robot for Students, Makers and Researchers". *Proceedings of the IROS workshop on Modular and Swarm Systems: from Nature to Robotics*, 2014.

(E) M. Pacheco, R. Fogh, H. Lund, and D. Christensen. "Fable II: Design of a Modular Robot for Creative Learning". *Proceedings of 2015 IEEE International Conference on Robotics and Automation*, 2015.

(F) M. Pacheco, H. H. Lund, and D. J. Christensen. "Fable I: A Modular Robotic Playware Platform". IEEE/ASME Transactions on Mechatronics, 2015 (Submitted).

(G) Patent Pending on "A Set Of Robotic Building Elements" Patent No. 16157134.4 - 1658.

Acknowledgments

This project would not have been possible without the support of any of the following persons. First I would like to thank my main supervisor Professor Henrik Hautop Lund and cosupervisor Associate Professor David Johan Christensen for their time and guidance during my Ph.D. For having always been prepared to discuss, brainstorm and offer guidance throughout many of this project's challenges. I would also like to thank Professor Radhika Nagpal and Senior Research Scientist Justin Werfel for their time, inspiration, dedication and guidance and for making me feel part of the Self Organizing Systems Research Group.

I would also like to thank my colleagues at DTU. Remus M. Prunescu for all of his help with Matlab, latex and control engineering. Finnur Smári Torfason for all of his support with KiCad and Python. Jari Due Jessen for always being willing to help, for sharing his ideas on play theory and for helping us get in contact with Coding Pirates. Dimitrios Papageorgiou, Andreas Søndergaard Pedersen and Mikkel Cornelius for the very well earned coffee breaks and Tania Groth for helping me proof read the document.

I would like to thank my family, both in Tijuana and La Paz, for their interest and all of their support. I would like to thank my loving wife Alondra Sofía Rodríguez Buelna for her incredible support throughout these three years, for allowing our home to be filled with robots, chips, 3D printers, late night prints and lastly for listening to me talk non stop about robots.

Table of Contents

Chapter 1

Introduction

We live surrounded by consumer products that constantly encourage us to purchase, but few that encourage us to build and to create. Taught to us from an early age, plagiarism and copyright policies serve as mental barricades that dry out our curiosity, creativity and collaboration [1]. Educational systems across the world are pushing to give children the necessary knowledge and tools that will allow future generations not only to be technologically competent but to awaken their curiosity towards developing greater and better tools for the world of tomorrow [8].

There are plenty of educational robots available in the market, where each of them proposes its own approach on learning and stimulation. Through our collaboration with Danish public schools and educational consultants we have learned that current platforms adapt poorly to the dynamics and classroom structure set by the Danish Ministry of Education, (e.g. lack of a quick setup time).

This project seeks to revitalize and quench users thirst for knowledge through the use of modular robotics. I believe that, given the right tools, anyone can become a robot designer. Throughout my Ph.D. I have focused on the design and development of Fable, a mechatronic construction kit that allows users to playfully build and program their own robotic creations. Through the inspirational power of robotics, Fable can help motivate users to learn complex topics like math or programming.

1.1 Background

The work of this Ph.D. project began with the dream of enabling everyone to be a robot designer and programmer. With this goal I began my search in the hope of answering the following questions:

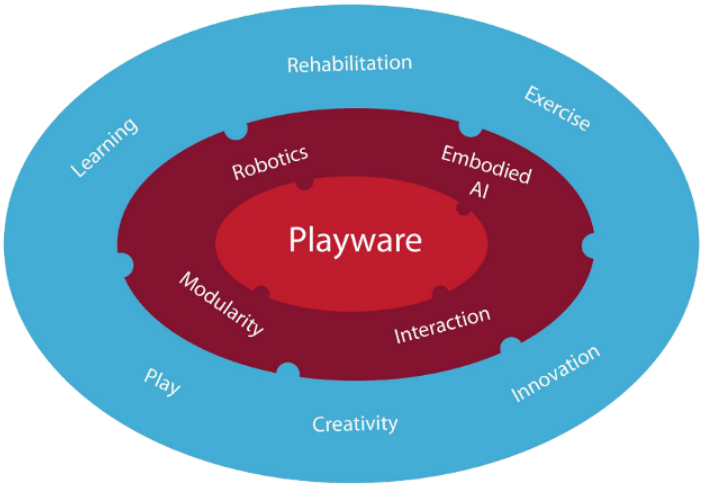


Figure 1.1: Playware's main focus fields (red) and the fields of application (blue).

- how can I design a robot that will allow both kids and adults to quickly build and design their own robots?
- How can I design a platform that will enable both experienced and non experienced programmers to engage in programming?
- How can we expand this platform in order to keep users motivated with new possibilities?
- What is the right set of modules that allows users to build a wide variety of robots?
- How can I design a system that uses state of the art modular robotics and still have a market oriented, price competitive system?

Center for Playware

Over the years digital technology has influenced and affected the way that we act, think, socialize and play to the extent of becoming part of our culture. Center for Playware is a cross-disciplinary research group, that aims at developing technologies capable of using a players deep focus within the mental state of play, for other non playlike activities such as learning, rehabilitation or exercise [9, 10].

By the term playware, we mean: *"[the] use of technology to create the kind of leisure activities we normally label play, i.e. intelligent hardware and software that aims at producing play and playful experiences among users and of which e.g. computer*

games are a sub-genre" [11, 12]. Our hypothesis is that if we combine knowledge of given user groups with recent technological developments in robotics, it would then be possible to create playful experiences out of boring and tedious ones, thus allowing users to engage in activities they would have otherwise not have found interesting enough to do.

Center for Playware makes use of robotics, modularity and interaction and embodied AI, in order to develop Playware technology and study its influence on its users. For my Ph.D. project I combined some of these fields with the hope to build a system that can motivate users towards learning topics like programming, problem solving and math while in a state of play. We hope that the work presented will nourish users' creativity and eventually push them towards innovating.

1.2 Objective

Ever so often our modern societies redefine themselves and search for new ways and new tools that can help prepare future generations for the world of tomorrow. In 2014 Denmark passed an educational reform to establish new learning objectives and working conditions in public schools [13]. This reform helps to push schools towards enabling close to 600,000 students to be technologically competent [14]. The research performed in this project is then to design a system that is capable of adapting well to the Danish educational system and potentially expand to other countries. Also to be able to adapt to various levels of experience, that is to serve as a tool for non-experienced users building hobby projects, and all the way up to supporting academics in the development of their research.

1.3 Contributions

The Ph.D. project on Modular Robotic Playware contributes to the robotic community in the following novelties:

- **Contribution 1:** The Fable System itself consists of a set of active and passive modules that can easily be programmed through the use of a visual programming language and has been tested with more than 500 school children.

- **Contribution 2:** Connector design: The connector contains magnets that allow a solid attachment and detachment between modules for rapid construction of robotic morphologies, which allows both kids and adults to quickly and with ease attach modules together. The connector design is scalable so users have the possibility to combine larger modules with smaller ones (see

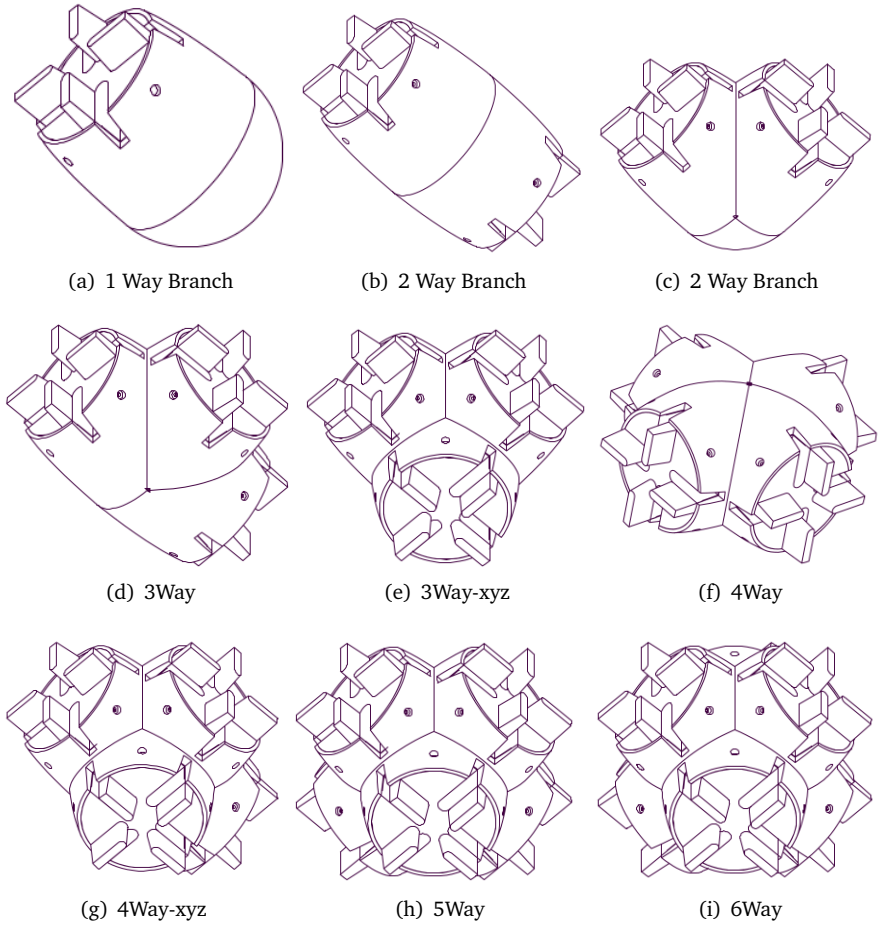
Fig. 1.3). This allows users to be able to use small robotic modules to mimic finer details in a robot, such as fingers while other larger modules could be used for a leg or torso. Each connector has a set of flanges and cavities that lock modules in place to prevent bending and twisting. Thus pulling modules apart is the only way of detachment. This feature makes it possible to build complex and sophisticated robots in a matter of seconds.

- **Contribution 3:** Scaffolding based progression in programming: The current design of the system helps introduce non experienced users to programming by using a graphical user interface based on Google's "blockly". Then when users are confident enough they can jump to Python and start programming more complex behaviours, plus eventually it can be compatible with Matlab and Simulink which could allow academics to perform state of the art research. Thus the system is able to adapt to various levels of complexity.

- **Contribution 4:** System Architecture. The system makes use of a USB dongle to send radio commands from the PC to any of the active modules. Users are able to target their communication to a specific set of modules by changing the radio channel used for messaging. Each of the 6 radio channels available is color coded so users can easily identify which modules RGB LED color matches the one on the dongle. This feature allows students at school to work on the same system while not receiving radio interference. Users are not require to compile their code to be able to run it on Fable, enabling the user to immediately receive feedback from their creations.

- **Contribution 5:** Building system that consists of a library of an 8 passive module architecture, see Fig. 1.2. This architecture allows users to build robots without any unused connectors, thus reducing the dead-weight of a robot assembly.

- **Contribution 6:** A set of active modules which includes a Joint Module, a Wheel Module and a Sensor Module. A Joint Module is a 2 DOF module that uses 2 the Dynaixel AX-12A motors. Furthermore this module has position feedback, control and speed feedback. Wheel Modules are modules that can be used to build robotic vehicles. A Sensor Module contains a 3 axis accelerometer, 3 axis gyroscope, 3 axis magnetometer and a distance sensor. All modules have their own LiPo battery, charging port, electronics, radio communication and RGB LED.



(a) 1 Way Branch (b) 2 Way Branch (c) 2 Way Branch

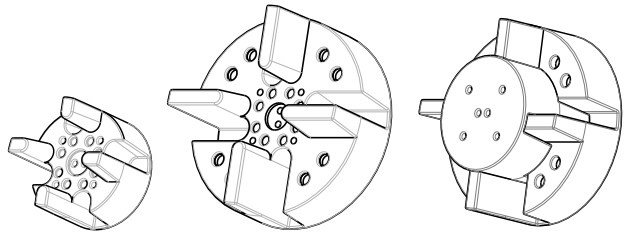
(d) 3Way (e) 3Way-xyz (f) 4Way

(g) 4Way-xyz (h) 5Way (i) 6Way

Figure 1.2: Passive Modules

1.4 Thesis Outline

The following parts of this thesis consist of a review of the state of the art within modular robotics in chapter 2, a field that has deeply influenced the course of this project. Furthermore in chapter 2 also presents thoughts on how such systems helped forge Fable's current design. Chapter 3 contains state of the art in educational robots which include commercial products as well as academic work and set the system's requirements. In chapter 4 I review the design methodology and rapid prototyping that helped us develop the various iterations of the modular robot. In chapter 5 the report goes through an overview of the design implementations made throughout the iterations, some of which are not documented in any of the



(a) Small connector (b) Medium connector (c) Compatibility between two sizes.

Figure 1.3: Connector design: With the current design any size is possible and compatible with the rest.

publications. Chapter 6 presents the tests and user cases that were implemented at various times at Center for Playware, at the Sciencetalenter in Sorø, Coding Pirates, Trekroner School, Antvorskovskole, HTX, Danmarks Læringsfestival and Big Bang Festival. Chapter 7 closes the document by presenting the conclusions and lessons learned throughout the development of the system. All of the publications I wrote as a part of my Ph.D. thesis, are listed in the appendices.

Chapter 2

Modular Robots

This chapter reviews the state of the art within modular robotics. Many of the systems presented have inspired the early and late development stages of Fable. Even though robots aligned under user-reconfigurable systems are more relevant to our approach, there has been a considerable amount of knowledge taken from self-reconfigurable systems, and for that reason, are included in this review.

For the purposes of this document I will consider modular robots as self-contained kinematic machines equipped with the capability of changing shape and functionality, on their own or with the aid of a user. A key advantage of modular systems is that they can adapt better to unknown problems than conventional robots [15]. By allowing the system's modularity to exploit its adaptability to solve tasks, one expands the reusability of the robot. On the other hand modular robots often perform poorly when compared to robots specifically designed to solve a limited amount of tasks in a known environment. The concept of modular robots was introduced by Fukuda in his work on cellular robots (CEBOT) [16, 17], where he envisioned a set of heterogeneous modules capable of disassembling to go through narrow cavities and reassemble, once it overcomes the obstacle, and solve a task. Many of the current modular systems tend to be homogeneous platforms consisting of several modules (cells) [18, 19, 20, 21], where each unit is equipped with proprioceptive and extereoceptive sensors, actuators and connectors allow modules to attach to their neighbors and form strong structural bonds.

While many modular robotic platforms have the capability to self-reconfigure [20] some platforms focus on a user's aid to reconfigure the system [22].This chapter will present an overview in chronological order of the state of the art in modular robotics by looking into two main categories: self-reconfigurable [23, 24, 25] and user-reconfigurable robots [26, 27].

2.1 Self-reconfigurable Robots (SRR)

Self-reconfiguring modular robots are robots equipped with active connecting mechanisms, thus giving the system the possibility of rearranging its modules [28]. The applications of self-reconfigurable systems range from space exploration [29], to search and rescue and some even explore the possibility of having adaptable furniture [30, 21].

The field of self-reconfigurable systems began with the work of Professor Fukuda in the late 80's with his paper on the Cellular Robot (CEBOT) [17]. CEBOT was mainly a theoretical work that explored the capabilities of shapeshifting systems. Throughout the years Fukuda's work has inspired many researchers in hopes of building and understanding the grand challenges of self-assembling machines.

It was until the late 90's when it became possible for researchers to build and experiment with systems that would later be known as self-reconfigurable robots. In 1998 a group of researchers at Dartmouth College published their work on a system called Molecule, where they showed how a set of modules could aggregate as active three-dimensional structures that could move and change shape. They also included algorithms for trajectory planning of modules [31, 32].

Two years later Mark Yim, one of the pioneers in the field, published his work on the Polybot [33, 34]. Yim had previously worked on other modular system's [35], he developed this robot during the late 90's at the Xerox Research Center in Palo Alto. The system was composed of 2 module types: segments and nodes. The first contains 1 DOF and two connectors, while the latter are rigid and contain six connectors. Though the system consisted of a limited variety of modules, it was still able to demonstrate its versatility by traversing through various terrains and by manipulating objects. Figure 2.1(a) shows one of the configurations of the system.

Later that year, researchers at the University of Southern California published their work on CONRO, a modular robot designed as a deployable system [36] targeted for tasks such as search and rescue and military surveillance. CONRO's capabilities where meant to outperform fixed-shaped robots when going through unexpected situations and environments. The system was able to reconfigure itself into a snake like robot or a hexapod. Each of its modules was self contained and equipped with its own battery, camera and antenna.

In 2002 AIST researchers Murata and Kurokawa published his work on the Modular Transformer (M-TRAN), which later became one of the most influential systems in its field. M-TRAN's unique mechanical design helps the system exploit many reconfiguration possibilities while using a homogeneous system. M-TRAN is capable of shapeshifting to functional rearrangements in a matter of seconds [37,

38]. The system has been successful in traversing environments in snake like form and change its shape into a quadruped to continue its journey, see Fig 2.1(b).

Two years later a group of researchers lead by Prof. Lund at the University of Southern Denmark published their work on a system called ATRON, see Fig 2.1(c). This robot was able to build a 3D rearrangement of modules by relying on a minimalistic approach of one degree of freedom per unit [19]. ATRON modules was the first self-reconfigurable system to reach a population of more than 100 units.

In 2005 the research project Molecubes, lead by Hod Lipson at Cornell University was made public [39, 29]. Molecubes uniqueness relied on demonstrating that robots are machines capable of autonomous self-reproduction. Each module consisted of electromechanical cubes able to attach and detach from one another by using electromagnets to selectively control the robot's shape. Three years later the group redesigned its platform for a different approach; to make molecubes an open source platform [40, 41] in the hope of getting more research labs, hobbyists and enthusiast to tackle some of the great challenges of self-reconfigurable robots [42]. The latest version of Molecubes focused on ease of manufacturability and low cost, plus it also gave users the possibility of expanding the system with their custom modules, shown on Fig. 2.1(d).

In 2006, researchers at the University of Southern California released their work on a new platform called Superbot [43], Fig. 2.1(e). Superbot was a second attempt towards fully functional deployable modular robot, CONRO being the first. Each Superbot module was equipped with 3 DOF and had a scale form factor similar to that of M-TRAN. Modules were equipped with, an on board battery but were also able to share power through their connectors. Superbot was designed to withstand harsher environments than most modular systems, therefore the connector design is moisture and shock resistant.

Connector Kinetic Robot or CKbot for short, was developed by Mark Yim's group at UPENN's Grasp Lab. CKbot, shown on Fig. 2.1(f), and gave a solution to the challenging problem of recovering after disassembly. A video attached to their publication in 2007, shows CKbot on a 3 module arrangement performing a walking gait that is then interrupted by a human and disassembled. After the disassembly, CKbot enters a phase of relocating its previous neighbors, reassembles itself and continues with its walking gait.

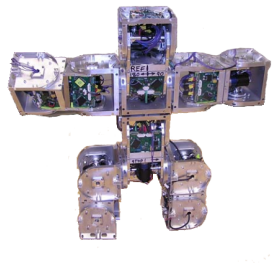
In today's modern societies, technology is being merged into everyday environments, ranging from smart watches to smart houses. With this in mind Ijspeert's group at the Swiss Federal Institute of Technology in Lausanne, (EPFL) wants to push this tendency further by designing modules capable of building self-reconfigurable furniture [21]. These modules called Roombots are equipped to build anything

(a) Polybot

(b) M-TRAN

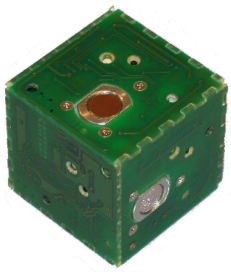
(c) ATRON

(d) Molecubes



(e) Superbot

(f) CKbot



(g) Miche

(h) Roombots

(i) M-Blocks

Figure 2.1: Self-reconfigurable robots.

ranging from tables to walls, from furniture like shelves to tangible or interactive roomware, shown in Fig. 2.1(h).

Meanwhile at MIT, the group of Daniela Rus was working on another approach to the reconfiguration search: reconfiguration through disassembly [44]. Their motivation relies on the reduced complexity of achieving a desired shape through disconnection, plus, due to the mechanical and computational limitations of most systems, a desired shape can be achieved much faster this way. Miche, see Fig. 2.1(g), was equipped with electropermanent magnets that allowed each cube-shaped module to change the polarity of its magnets in order to detach itself from its neighbors.

In 2011 MIT researchers published their work on Motorized Proteins (Moteins). Moteins are foldable 1D strings that can be programmed to self-assemble in 2D and 3D shapes. Each string is seen as a very simple robotic module [45], where each module has between 1 and 2 DOFs and simple actuators with one or two states. In their work, the researchers made a mathematical proof that Moteins could produce any polygon.

The last addition to the self-reconfigurable robot ecosystem is M-Blocks [18]. These colored cubes, released from Daniela Rus' lab at MIT, present an alternative to traditional self-reconfiguration proposals from earlier systems like M-TRAN or Polybot. M-Blocks' innovation comes from the use of the inertia momentum, generated within the module, to flip from one place to another. In recent publications M-Blocks have demonstrated the accuracy and versatility that can be achieved while still having a module with a minimalistic design, see Fig. 2.1(i).

Even though there has been extensive work on self-reconfigurable systems there are still many problems to be solved, including a time and energy efficient way towards self-reconfiguration for any of the presented modular systems. While this field has struggled to solve some of its challenges [15], solutions to similar problems have been found in other fields such as swarm robotics [46], and maybe origami robots [47, 48] could shed some light on other difficulties within self-reconfiguration.

2.2 User-reconfigurable Robots (URR)

User-reconfigurable systems take advantage of the limitations of today's self-reconfigurable capabilities and use it to exploit users curiosity and creativity. Good examples of this include systems that have a programming-by-building [49] approach, or systems that make use of programming-by-demonstration [50]. Other platforms use more conventional ways of programming while allowing users to ex-

ploit the system's potential such as building a CNC machine with LEGO Mindstorms [51]. As we will see, researchers and companies have had a higher success rate in taking ideas from this field out of the lab and building successful products out of them.

The first and most iconic success story in user-reconfigurable robots is Lego Mindstorms. Mindstorms, as its name suggests, takes inspiration from Seymour Papert's work [52] and is currently the most well known robotic kit on the market. Mindstorms started as a collaboration between the MIT Media Lab and Lego. The product was launched in 1998 as a hobby robotic construction kit for users from ages 12 and up. Shortly after it was introduced into schools, and since then it has been used in beginner courses in programming to advanced courses in operating systems, compilers, networks and artificial intelligence [53]. The Mindstorms kit is based on Lego Technic parts, plus a set of sensors and actuators that can be controlled using a programmable controller box. Users can program the system through a graphical programming interface that allows them to drag and drop blocks to easily program their robots behavior. Figure 2.2(a) shows the standard Lego Mindstorms kit which retails for $349.99.

In 2003 Professor Henrik H. Lund, from the University of Southern Denmark, published his work on the I-Blocks [54, 55, 56]. I-Blocks were a set of intelligent Lego Duplo bricks that were packed with a microcontroller and sensors. With this system kids were able to program complex behaviours by building with smart Lego bricks.

Topobo, see Fig. 2.2(b) began as a research project [57] and, as many others, relies on two types of components: passive and active. Topobo lacks external sensors, thus the only active module is an actuator. However, the innovation of the system lies on the way that users are able to interact with it. Users can program their robots by demonstrating the sequence of movements that they want their system to execute. To do this they set a motor into teach mode and are then able to record the movements that they want the system to repeat. This approach of programming by demonstration has been previously used in industrial applications but not as a consumer product for kids, even though similar attempts were done earlier by other toy companies like LEGO [58].

In the early 2000's Robotis, a South Korean company, launched The Bioloid kit. Bioloid is a hobbyist and educational robot kit with a structure similar to the one used by Lego Mindstorms. Bioloid, shown in Figure 2.2(c), is a centralized system based on brackets actuators, sensors and a control box. The uniqueness of the system relies on the design of its actuators, the Dynamixel AX-12A series which are higher torque servos with feedback (e.g. position, torque, speed). Bioloid allows

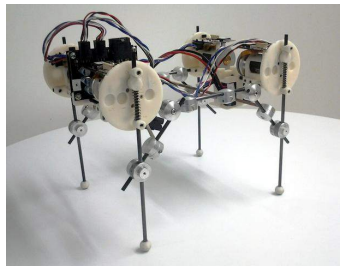
(a) Mindstorms

(b) Topobo

(c) Bioloid

(d) Odin

(e) Cubelets



(f) Locokit

(g) Myorobotics

(h) MOSS

Figure 2.2: User-reconfigurable robots.

researchers and hobbyists to build their own robots and program them using C++.

In 2006, the Playful Invention Company released PicoCricket, a system that allowed kids to plug lights, motors, sensors, and other devices to build interactive environments. Kids can for example make a cat and program it to purr when someone pets it, or you could also make a birthday cake and have it play a song when someone blows out the candles. A key difference between this system and LEGO Mindstorms is that instead of users making LEGO robots, they can build interactive artistic projects by using a wider range of materials, like pipe cleaners.

Odin [59] is a biologically inspired system that explores the idea of having deformable modules in a lattice structure. The system was developed in the late 2000's at the University of Southern Denmark, by some of the same researchers that worked in the development of ATRON. Odin's uniqueness lies on modules attaching through flexible connection mechanisms, an arrangement of modules is shown on Fig 2.2(d).

Cubelets released as a product in 2011, originally called Roblocks, started as Schweikardt's Ph.D. project at Carnegie Mellon University, [60, 61]. Cubelets is a set of heterogeneous cubic robots that when users snap them together, behaviours emerge. Even though there is a logic while building with Cubelets, every now and then unexpected behaviours are obtained, which help as a hook to attract users' attention tor try and decipher the logic behind a given behaviour. The system consists of a centralized power cube, sensor cubes (e.g. distance sensor, actuator cubes (e.g. wheels, rotary motor), and interface cubes (e.g. knobs, and displays). This system is currently a commercially available product owned by Modular Robotics that retails for $159.99 for a 6 robot kit. Figure 2.2(e) shows a construction made out of 13 cubes.

Locokit is a robotic kit inspired by artist Theo Jansen's work on kinetic sculptures. Locokit was developed at the University of Southern Denmark[27]. The system, shown in Fig. 2.2(f), is meant to ease the pain of building robots for research. Users assemble their robots using a set of rods, springs, electronics and Dynamixel motors and they are able to program their robots through a web browser.

EZ Robot is a Canadian company with a focus on selling minimalistic robotic modules that allow users to create their own personalized robots. The modules clip on to each other where most modules consist of passive parts, servos, lights or sensors. Users can then interact with their creation using a PC or tablet.

In 2013, the Myorobotics system is designed for researchers as a toolkit that supports them in building prototypes to be able to implement biologically inspired algorithms such as in the field of neuroscience [26]. This is done through the use of cable-driven actuators together with a design that allows a combination

of lightweight, high strength and relatively compact designs. An example of the Myorobotics platform is shown in Figure 2.2(g).

Moss is the newest addition to the Modular Robotics family and it differs from their previous release, Cubelets, in allowing users to build more things by combining passive and active building blocks. The uniqueness of Moss comes from the use of magnetic spheres with which to create both strong attachment between neighboring blocks but also weaker bonds like hinges. The price per Moss module ranges from $19.95 to $59.95, depending on the functionality.

2.3 Inspiration taken from the State of the art

This chapter reviewed the state of the art in modular robotics. The systems that were mentioned have influenced Fable's high level design, guided by the design objectives from chapter 1 as well as by the system's requirements that will be presented in the next chapter, by helping us decide whether to choose to build a system based on self-reconfigurability or user-reconfigurability, whether to implement neighbor to neighbor communication or search for alternatives, whether to design a homogeneous system or a heterogeneous system, whether to choose between centralized control or distributed control and whether it is necessary to commit to a specific module size or experiment with several. The following sections will explain the pros and cons of each of the alternatives and which one was chosen for Fable.

Self-reconfigurability vs. user-reconfigurability

Thanks to Center for Playware's direct experience in the development of the self-reconfigurable robot and their vast knowledge on the challenges and limitations of building a SRR, we decided on developing a system that would rely on users to reconfigure it. The SRR research field has its challenges such as finding a solution and achieving it from any begin state to any goal configuration with any population of modules [62]. However through the design of a user reconfigurable system we have the possibility of turning a challenge into an opportunity to stimulate users' creativity, which is why we have considered user-reconfigurability as an advantage for Fable.

Neighbor to neighbor communication vs. alternatives

Most of the systems presented under SRR contain connectors that are capable of transmitting electrical signals either to share communication or to share power

or both. In this approach connectors must contain electronics to manage the communication and power being transferred between modules, which tends to overcomplicate manufacturability and implementation, increase production costs and jeopardize the reliability of the system. In contrast by avoiding transferring signals through connectors modules will have to be charged individually and the system will not be able to determine its own topology. During the early development stages of Fable [3] there was an attempt to implement neighbor to neighbor communication but was later disregarded due to the disadvantages just mentioned.

Homogeneous system vs. heterogeneous system

Self-reconfigurable robots tend to be homogeneous, an approach that in one sense simplifies the self-reconfiguration process by having all modules contain the same advantages and limitations. But in another sense increases cost by having all modules contain the same sensors, battery and active connectors. However user-reconfigurable robots tend to go for a heterogeneous approach because it helps to reduce costs, complexity and exploits users creativity by allowing them to build various robots by combining an assortment of modules. Thus the advantages of having a homogeneous system are very limited when one is designing a user-reconfigurable robot and for this reason we decided to build a heterogeneous system.

Centralized Control vs. Distributed Control

While most SRR are distributed systems this solution tends to increase complexity both in manufacturing and in the development of software, which also tends to overcomplicate programming of the system on the user's side, since it is more complicated to develop a solution for a distributed system than a centralized one. A distributed solution also increases the requirements of computational power per unit which ends up increasing the cost per module. On the other hand a centralized control outsources the computational demand to third party devices like a PC or tablet, leaving minimal computational operations to be executed locally by each module. Thus a centralized approach reduces production costs and increases the user friendliness of the system, reasons why we decided to implement this approach for the high level control while allowing active modules to perform some of the computations locally.

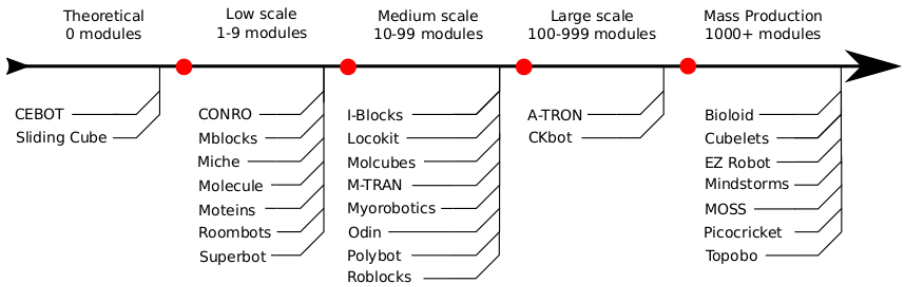


Figure 2.3: Diagram displaying the manufacturability scale that each project reached. From left to right first are the theoretical systems, i.e. systems with no prototypes, followed by low scale production of modules, that is between 1 and 9 module population. Afterwards comes the medium scale production of modules, which includes populations of 10 to 99 units and the large scale production encapsulates systems that have produced between 100 to 999 units. The last scale is mass production with more than 1000 units.

Fixed module size vs. various sizes

Most SSR and URR tend to be compact devices. There is after all an inherited need to use several modules to build a given robot configuration. Reducing the size in modules has its disadvantages when it comes to the quality of actuators, batteries, sensors, some of which are either bulky or power hungry. However a small form factor also has the advantage of allowing users to build robots with finer details, e.g. a robot hand with actuated fingers. Large modules tend to be bulky and more difficult for small kids to handle but they have the capability of housing more powerful equipment. However by combining module sizes it is possible to exploit both advantages, for this reason we decided to explore if there was a middle-ground that will allow us to have modules of various sizes.

2.4 Closing thoughts

Through Fable I want to democratize robot building, so unlike most of the systems presented whose purpose was purely research oriented, there is a need for a mass producible system. To achieve this goal it is necessary to move away from three key features commonly overlooked during the development of research prototypes which are:

- Reduce manufacturability costs, that is to cut away nonessential features that will increase costs without a clear benefit for the user (e.g. neighbor to

neighbor communication).

- Reduce complexity of the system to ease manufacturing, assembly, use and repairing of modules.
- Increase reliability and usability to ensure that a module can be used by non-expert users in everyday situations.

Historically user reconfigurable systems have had greater success in porting ideas onto the consumer market. We believe that by taking inspiration from the systems mentioned in section 2.1 - 2.2 and by focusing on the three key points mentioned before we will be able to produce a modular robot system that would fit under the large scale production of modules shown in Fig. 2.3.

Chapter 3

Educational Robots

A major illusion on which the school system rests is that most learning is the result of teaching. Teaching, it is true, may contribute to certain kinds of learning under certain circumstances. But most people acquire most of their knowledge outside school...

Ivan Illich, Deschooling Society

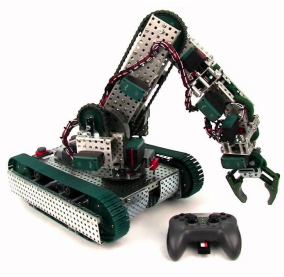
This chapter presents a state of the art of educational robots. Due to the nature of research fields some platforms fit both in the review on modular robots and in this chapter, I will, for the sake of clarity, describe those cases in both chapters.

3.1 Background

Educational robots are here whith the hope of revolutionizing the way kids are taught in schools and the way we learn about new topics. In the late 60's to early 70's, american intellectuals such as the Swiss psychologist Jean Piaget [63], Paul Goodman [1] and Ivan Illich [2] began to strongly criticize the way schools taught children. Influenced by this new wave of thinking, professor Seymour Papert created the Logo programming language, together with his colleagues Daniel G. Bobrow, Wally Feurzeig, and Cynthia Solomon. The name Logo derived from the greek word logos, meaning thought. Logo is mostly known for allowing kids to program a turtle that produced line graphics by driving around with a small retractable pen.

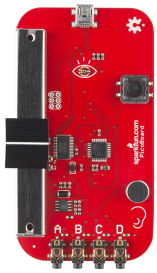
Through the creation of Logo and his thoughts on education (i.e. constructionism [64]) Papert paved the road that for the development of educational robots. Platforms range from very minimalistic robots that allow kids to build up experience on cause and effect through story telling, drawing, to more complex systems that

(a) Mindstorms



(b) VEX ED3

(c) Nao

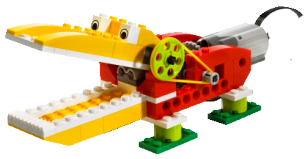


(d) PicoBoard

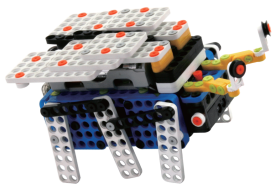
(e) iRobot Create

(f) Pleo rb

(g) Thymio II



(h) WeDo



(i) Ollo

Figure 3.1: Educational robots.

enable adults to learn about programming. The field of educational robots has been growing dramatically for the past couple of years, thus for the sake of brevity I will only go through some of the most popular systems used in schools, as well as other less known but nonetheless interesting and commercially available systems.

3.2 Overview of systems currently available

LEGO Mindstorms [65] is the most iconic robotic system for education. LEGO branded the platform drawing inspiration from Seymour Papert's work on education. The system was developed in the early nineties in collaboration with MIT Media Lab and it has proven to be the leading robotic kit sold for education today. There are numerous competitions using this set, amongst which are the First LEGO League, RoboCup Junior and the World Robot Olympiad. LEGO Mindstorms is a robotic construction kit based on standard LEGO Technic parts and a programmable brick on which users can plug in sensors and actuators. The kit comes with software that allows users to program their creations through block programming, Fig. 3.1(a) shows an image of the standard LEGO Mindstorms kit.

Years after the 1998 release of LEGO Mindstorms, VEX opened their branch called VEX EDR [66], shown in Fig. 3.1(b). Since then VEX Robotics has become one of the leading systems in education and one of LEGO Mindstorms biggest competitors. Their system, although with a different design, is very similar to the one offered by LEGO. Users can connect sensors and actuators onto the programmable block and use standard building parts to shape their creations. Just as LEGO, VEX offers international robotic competitions where students obtain experience in problem solving as well as international recognition for their achievements.

In 2005, Aldebaran released their flagship product called Nao [67]. Nao is a humanoid robot that is 58 cm tall and capable of highly sophisticated movements. Equipped with a camera, microphones and speakers, teachers are able to use Nao as a teaching assistant. Nao can potentially help students learn, for example, multiplication tables, through a series of repetitive pronunciations and by acknowledging when a user has given the right input. It has also been used by teachers and psychologists working with autistic children [68]. Children with autism could find it easier to interact with Nao than with a human adult, thus Nao could play the role of an intermediary between the teacher/psychologist and the child. Fig. 3.1(c) shows a picture of this humanoid robot.

A year later, in 2006, the Playful Invention Company released PicoCricket [69], a system that allows users to plug in lights, motors, sensors and program them to react to a given scenario. An example could be to make a birthday cake and have it play a song when someone blows out the candles. PicoCricket encourages users to build interactive artistic projects and environments, Fig. 3.1(d), shows a picture of the electronics board.

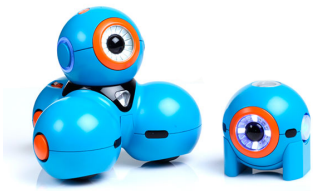
In 2007 iRobot, creators of the Roomba, released their platform called Create [70]. This robot, currently in its second release, takes the design of the already

(a) Cubelets

(b) Sphero

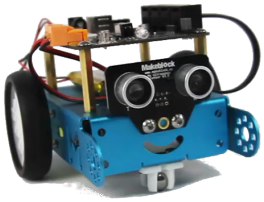
(c) Linkbot

(d) Ollie



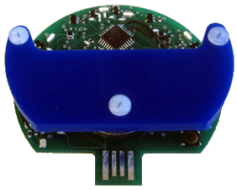
(e) Dash and Dot

(f) Romibo



(g) mbot

(h) Ozobot



(i) AERobot

Figure 3.2: Educational robots.

popular vacuuming robot and makes it compatible with development platforms like Raspberry Pi and Arduino. The iRobot Create has basically removed the cleaning mechanism of a Roomba and left a programmable differential drive robot, as seen in Fig. 3.1(e), on board batteries and basic sensors such as odometry and a bumper for students to start programming.

Later that year the world was introduced to Pleo rb, an animatronic pet dinosaur whose behaviour matures and changes with time. Currently manufactured by Innvo Labs but originally developed by the creators of the well known Furby, Pleo, shown

in Fig. 3.1(f), has the look and feel of a week old Camarasaurus dinosaur [71]. According to Innvo Labs, Pleo is not your ordinary robot, as it is able to "learn" from its experiences and environment through the use of artificial intelligence and eventually develop an individual personality. Pleo rb can recognise colors, patterns and drop-offs so it reduces the risk of it falling from a ledge. It can hear, and turn towards the sound source. It can sense the temperature of its surroundings and react to its environment. It can distinguish from a gentle touch or a hit. An internal clock allows it to determine the time to wake up, eat and sleep.

Thymio [72], see Fig. 3.1(g), is a differential drive robot developed in 2008 at the Swiss Federal Institute of Technology in Lausanne. The platform comes with a set of sensors that allow the robot to interact based on light and touch. A set of LEGO knobs and sockets on the top and sides of the robot allows users to expand the system using standard LEGO bricks. Thymio comes with its own graphical programming environment that helps non experienced programmers get started while more experienced users can program the system using a script language.

Later in 2008 LEGO released a new educational product called WeDo [73], shown in Fig. 3.1(h). This product is an easy-to-use construction set by LEGO that helps jumpstart young students into the field of robotics. The system includes a LEGO Education WeDo Software and Activity Pack. With this system students are able to build standard LEGO models and add working motors and sensors to them and to program them. LEGO claims that students are able to explore a series of cross-curricular, theme-based activities while developing their skills in science, technology, engineering, mathematics and even social skills such as language and literacy. The WeDo set includes standard LEGO bricks and a special block like a motor, tilt sensor, motion sensor, and LEGO USB Hub.

OLLO by Robotis is a construction set that introduces kids to robotics [74]. OLLO consists of a set of kits that users can combine to build more ambitious robots, see Fig. 3.1(i). Young students can program their creations using Robotis' programming environment Robo Plus, also used in other systems like Bioloid. By using this platform the company advertises that kids, are able to learn about math, physics, programming, among other fields.

Cubelets is a modular robotic platform that as the name suggests, is made out of cubic modules, see Fig. 3.2(a) [49]. Each module is packed with electronics and six magnetic connectors that allow power to flow through them. Module's are color coded according to functionality in the hope of inviting users to mix them and see behaviours emerge from their creations. Cubelets are used in an educational environment in order to teach children about coputational thinking without the use of a written or graphical programming language.

Sphero, shown in Fig. 3.2(b) is a robot produced by a company of the same name [75]. This pocket size spherically shaped robot, as hinted by its name, can be controlled through a smartphone or tablet. Sphero comes with a variety of apps that users can download to start interacting with Sphero, ranging from remote controlling the robot, to augmented reality apps that allow the user to treat Sphero as a virtual pet. Sphero is currently used in educational environments to teach about graphs, speed, orientation in a map, among other things [76].

Zeno's open-platform software allows users to customize its facial features, and program the robot to speak in 26 languages [77]. The robot is able to carry on simple conversations and show compassion. Pedagogues can use Zeno as an intermediary with which to interact with autistic children. Users can remote control Zeno and have it interact with kids, answer questions and give short lectures on varied topics like astronomy, sports or films. Zeno's application is similar to that given to Nao robots, although Zeno's ability to express feelings through facial gestures could make it more suitable for children with autism than Nao [78].

EZ Robot is a company with focus on selling minimalistic robotic modules that allow users to create their own personalized robots. The modules are clip-on to each other where most modules rely on either passive parts, servos, lights or sensors. Users can then interact with their creation using a PC or tablet.

Linkbot, see Fig. 3.2(c), began as a Kickstarter project and was recently launched as a product by Barobo [79]. Users can interact with the system immediately out-of-the-box by using a feature called bump-connect. For those who are intimidated by programming Barobo offers a feature called PoseTeach which is something similar to programming by demonstration. Each robot is equipped with two rotating connectors that serve as wheels, an on board battery, a 3 axis accelerometer, an RGB LED and a buzzer, Zigbee, capable up to 100 meter range, plus users are able to attach many other things to Linkbots by making use of the M6 screw holes on the connectors. Linkbots can be programed in C, Python or the programming environment that comes with the system. Each module is packed with an I2C bus that expands the system's capabilities by allowing users to attach of-the-shelf sensors from third party developers.

Ollie [80], see Fig. 3.2(d), is the second release of the company behind Sphero. It is a fast rolling cylindrically shaped robot that allows users to make it do tricks as well as to customize its looks through accesories. Currently its educational use is similar to the one given to its older brother Sphero, with the advantage that Ollie's current retail price is lower.

Dash and Dot, shown in Fig 3.2(e) are a set of robots that help kids learn about programming and problem solving [81]. Dash is a three wheeled robot that can

be programmed to do basic movements, obstacle detection and complex sequences of instructions. Dot can detect whenever it is being moved. Kids can make use of accessories to expand the system's capabilities. Dash and Dot come with apps that help kids interact with them, play with them and program them.

Romibo is a robot that drives, tracks peoples eye movements and speaks 26 languages [82]. It is a platform used to teach and interact with children with autism. Therapists use Romibo as a tool to engage children in social interactions. Romibo is covered in colored fur, to make it appealing and friendly, plus a screen that serves to display the robots emotions. Users can use apps to program and remote control it.

MBot, see Fig. 3.2(g), is a robot meant for kids to enjoy the hands-on experience about programming, electronics, and robotics [83]. Mbot is a differential drive robot that interfaces with Scratch 2.0. The mainboard of mbot contains a buzzer, an RBG LED, a button, a light sensor, a motor port and an IR module. With Mbot kids are able to learn about science, technology, engineering and math.

AERobot (Affordable Educational Robot) is a lowcost educational robot developed by Mike Rubenstein from the Self Organizing Systems Research Group, lead by Prof. Radhika Nagpal, part of the Wyss Institute of Biological Engineering at Harvard University [84]. AERobot, shown in Fig. 3.2(i), claims a low cost production of $10 per unit, considering a minimum lot of 1,000 units. Some unique features of the system have allowed it to be considerably cheaper, compared to similar solutions, like Thymio II. Some of those design decisions include a PCB that serves both as a standard electronic board but which also gives structural support to the robot. Other design features of the system include the built-in PCB USB connector and the substitution of actuators with vibrators.

Earlier this year Ozobot was released, shown in Fig. 3.2(h), a tiny wheeled robot that reacts to colors it sees while driving on a piece of paper or screen [85]. Users don't program the robot. Instead they are able to influence its behaviour by triggering preprogrammed actions through color code combinations of red, green, blue or black that the robot can read while driving over them. With the use, of the four apps that are currently available, users can start using their Ozobots to race, to lure them to a given goal and thus teach users about programming through drawing.

3.3 Danish School Reform

In 2014 the Danish Ministry of Education passed a school reform that affected all public schools in Denmark. Danish public schools include the Danish municipal primary and lower secondary school and consists of a compulsory pre-school

class, from grades 1 to 9 with an optional 10th grade. The Danish schools have approximately a total of 600,000 students affected by this reform, where one of the changes stipulated is longer school days [14]. The new school hours per grade are the following:

- 30 hours a week from pre-school to 3rd grade.
- 33 hours a week from 4th to 6th grade.
- 35 hours a week from 7th to 9th grade.

Furthermore this reform pushes schools to allocate more time and opportunities for practical and application-oriented forms of teaching that will promote innovation, entrepreneurship and creativity. The reform also states that there will get an increase in the number of math lessons from 4th to 9th grade by one additional lesson per week, plus there will also be an increase in the number of lessons in natural sciences/technology from 2nd to 4th grade by one additional lesson per week. There will also be an introduction to elective courses from 7th grade, these courses include technology workshops. Students will also be able to choose elective subject packages featuring various themes such as innovation and natural sciences.

In addition the required number of teaching has increased to 41 working hours per week, out of which approximately 19 are teaching hours [86, 87]. All of these new changes have put more work load on Danish school teachers. Furthermore schools are expected to teach students the best skills and to nurture their innovation, entrepreneurship and creativity under a constant budget. That is most schools will not be able to afford expensive equipment, so if a school decides to buy an educational tool it is important that the tool fits well within the educational environment. The following list gathers the most critical points an educational tool should fulfill.

Requirements

- The system must have a competitive price so most schools can afford it.
- The use of the system must fit within a 45 minute lecture.
- Teachers need to have access to relevant educational material in Danish.
- It has to be easy to setup, e.g. less than a minute for a non-technical user.
- It has to be easy to pack up. Teachers have to verify that all of the robot parts were returned.

- Same tool for several school grades.
- Same tool for several topics.
- The learning must be in line with the official learning objectives set by the Danish Ministry of Education.

Robot	Robot Type	Parts	Grade	Programming Style
Dash and Dot	Monolithic	2	2nd	Block programming
EZ-robot	Modular	13	8th	Script language
Mindstorms	Modular	550+	5th	Block programming.
mbot	Modular	50+	2nd	Block programming.
Nao	Monolithic	1	7th	Script language.
Ollie	Monolithic	1	3rd	Block programming.
Sphero	Monolithic	1	3rd	Block programming.
VEX ED3	Modular	300+	6th	Block programming.
WeDo	Modular	150+	2nd	Block programming.

Table 3.1: Comparison of the most common educational robots used in Danish public schools. The Grade column represents the earliest school grade suitable for each platform, these grades were determined by Eva P. Christensen a pedagogical consultant from the University College Zealand.

In this chapter we have seen an overview of the currently available educational robots as well as some of their advantages and claims. Table 3.1 shows a list of systems in alphabetical order that are currently being used in Danish public schools. The systems in this Table will serve as an inspiration during the development of Fable and as a comparison for the results obtained by my design. Systems like LEGO Mindstorms, VEX ED3 and WeDo are meant to be used to build robots that are composed of many small parts. This presents an inconvenience for teachers since they have to count to make sure that every kit is complete after using a system in a lecture. Another aspect is that these systems brand themselves as being construction systems that allow kids to build a variety of robots and learn from the building process, however it is difficult to build and test a robot with such a system and fit it into a 45 minute lecture. As a result most teachers tend to build the robots themselves, never change them and align their learning objectives to fit the robot they already built.

Sphero, Ollie and Dash and Dot are systems that have the advantage of a quick setup time and are robust and simple to use. However the systems like Ollie and Sphero are quite limited in an educational environment, mainly due to the fact that they were specifically designed as entertainment devices for the consumer market.

These systems are programmable through a third party app called Tickle which allows users to program various systems by using Scratch.

EZ-robot is another system that was designed as an entertainment robot. As a result teachers tend to find it complicated to use since it's target audience are mainly hobbyists and makers. Furthermore the quality of the servos is low, schools have told us that they tend to burn often. Since the system was designed for tech hobbyists most teachers find it complicated to replace motors or parts in the system. EZ-robot is a system that even though it is modular most of the parts are very unique for each kit so schools need to buy several kits in order for kids to build their own robots.

Mbot is a robot that is affordable and simple to use. It is a modular system that relies on users buying different construction kits and mix them to begin to build their own robots. However due to the time restrictions most schools do not bother to change the robot's configuration. The system's ability to interface with Scratch makes it a very good tool in Danish schools. The system that has limited capabilities so it would not be able to adapt well to various school topics or class grades.

Nao is an expensive robot that costs around 60,000 DKK, which exceeds most of the public school's yearly budgets. It also adapts poorly to the lower class grades (6th and lower) in public schools. We should not forget that Nao is also a sophisticated robot that most teachers do not know how to program or operate.

3.4 Summary

By learning from the systems currently used in Danish public schools as well as from the teachers' demands, we will aim at designing a system that will fulfill the latest needs of the Danish public schools as specified by the Danish Ministry of Education. To measure our progress I have presented, in section 3.3, a list of requirements that an educational robot must fulfill in order to successfully adapt to the Danish educational sector.

Chapter 4

Production Tools

This chapter describes the tools and methods used throughout the development of Fable prototypes. The chapter will first go through a brief introduction to the rapid prototyping methodology and afterwards it will go over various types of tools that where used such as 3D printing, casting and vacuum forming.

4.1 Rapid prototyping: Tools and Methods

Rapid prototyping is an experimental approach towards system development, it is best suited when the design or behaviour of a system is not fully understood. We, at Center for Playware, use this method as an approach to produce robotic modules quicker than by using conventional methods for early testing and adaptation.

As described by the Standard Guide for Rapid Prototyping of Information Systems, a set of tests throughout various system iterations has helped narrow down the design to one that aligns better with the danish educational environment [88]. Through this process prototypes have helped us as developers understand the requirements in an educational environment. This in turn allowed us to test the design and functionality of the system. Throughout several iterations some of our designs have evolved into operational modules, e.g. Joint Module and Passive Modules. Other prototypes have helped us to explore the feasibility of a new idea or design that will eventually be incorporated to the Fable system, e.g. Camera Module.

An advantage of rapid prototyping is that it is a quick way to produce several iterations of the prototype. However in most cases the overall time required for system development is equivalent to the time it takes to develop with conventional methods [88].

Throughout this process quality is sometimes sacrificed to give way to a faster

production of prototype generations, though aspects such as low cost, robustness, efficiency, portability among other things have to be addressed for the final implementation of the system.

4.2 Additive Manufacturing

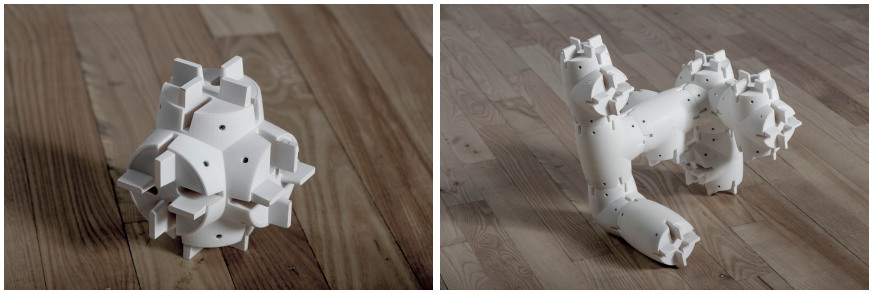
3D printers have risen in popularity in the past years. Slowly moving away from unreliable machines found mainly in hobby/maker sheds to sturdier designs made for home users. Throughout the development of Fable's mechanical parts I made use of two 3D printing technologies: Fuse Deposition Modeling (FDM) and Selective Laser Syntering (SLS).

Fused Deposition Modeling FDM

Fused deposition modeling (FDM) is an additive manufacturing technology commonly used for prototyping. FDM is better known as the method used by most DIY 3D printing kits like RepRap, Makerbot or Ultimaker. The technology consists of a die that heats up to a temperature high enough to melt a given material, usually polylactic acid (PLA) or acrylonitrile butadiene styrene (ABS). When the desired temperature is reached, a motor pushes the material through the die and deposits a first layer of the shape to be printed onto a platform. This means that if the shape to be printed is a cube then the first layer would consist of a square. By the time that the die has deposited the last drop of material in one layer, either the bed or the die retracts a distance equivalent to the layer thickness and the process repeats itself until the desired 3D object is obtained.

(a) 3D Touch (b) Cubify

Figure 4.1: FDM printers used in early prototypes.



(a) Passive Module (b) Set of Passive Modules

Figure 4.2: SLS Prototypes from Shapeways.

PLA is most commonly used material, because its melting point is much lower than ABS and its cooling properties are much more forgiving to this technology. ABS tends to crack layers apart with fast cooling, and it is also a material that has difficulty to adhering to the printing surface. Solutions to this problem include the use of liquids, made of a mixture of ABS and acetone, that the operator has to put on the printing surface before the print job starts. The 3D printers used during the early stages of prototyping include the ones shown in Fig. 4.1.

Selective Laser Sintering (SLS)

Selective Laser Sintering (SLS) printers consist of a set of rollers that deposit a thin layer of polymer dust on a chamber. A laser then aims its ray to the dust and melts together all of the area contained in a cross-sectional cut of the model to be reproduced. Once this process has finished, the chamber retracts a small amount and the rollers deposit another layer of fine dust into the chamber. The process then repeats until the printed object is complete. At the end of the job the object is buried in a cube of dust and has to be dug out and dusted with an air gun. SLS is much more accurate than FDM and by making use of third party printing services we were able to produce parts at a lower cost than by using our own printers. Most of Fable's passive modules were produced using this technology, See Fig 4.2.

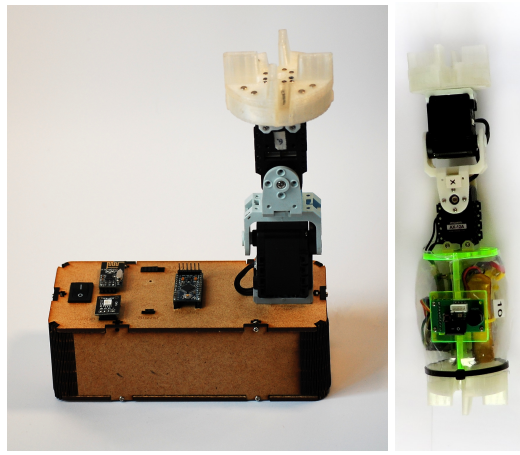
4.3 Subtractive Manufacturing

Subtractive Manufacturing refers to manufacturing procedures that rely on the removal of material to achieve the desired product, examples of it include milling or turning.

Laser Cutter

This is a technology that makes use of a high powered laser beam to cut materials, such as wood, metal or acrylic. Thanks to the new low-cost laser cutters available in the market, laser cutting has become popular among schools, small workshops and maker spaces.

Most laser cutters have a motion control system that interprets a G-code of the pattern that should be cut out of the material. Most commercially available laser cutters focus on cutting flat-sheet material, although there's current research on trying to expand the systems capabilities with a low cost investment [89]. Laser cutting was used to produce the early prototypes, see Fig. 4.3.



(a) Iteration 1 (b) Iteration 2

Figure 4.3: Joint Modules cut with the Epilog laser cutter from the SkyLab.

Computerized Numerical Control Machining

Computerized Numerical Control mills make use of high precision controls and calculations from a computer in order to cut various materials ranging from soft plastics to metals. CNCs make use of a programming language called G-code, although some brands make use of their own copyrighted versions. CNC mills include many functions like face and shoulder milling, tapping, drilling and some even offer turning. Standard CNC mills have 3 axes but modern models can go up to 6 axes. When we began producing iteration 3 of the Fable system, we placed an order of 130 connectors at a CNC manufacturing facility in China. The connectors where produced out of solid ABS blocks which made them significantly hevier than

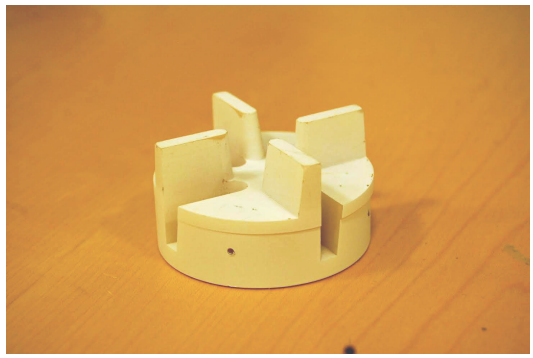
the 3D printed versions. The connectors were made using natural color ABS which then had to be painted white in order to match the color of the system. Figure 4.3 shows one of the milled connectors. For such works a STEP or IGES file is sent and the manufacturer, who then develops the G-Code for his machine. Even though the quality of the connector was high, the disadvantages mentioned earlier made SLS a better candidate.

4.4 Vacuum Forming

Vacuum forming consists of a sheet of plastic that is heated to a forming temperature, then stretched onto a single-surface mold, and forced against the mold by vacuum. Most machines contain a heating element on the top part where a frame holds a sheet of plastic close to the heat. When the plastic reaches its melting point the sheet will bend down. Then an operator will bring the frame with the sheet of plastic down to the base of the machine where a mold is in place. Vacuum holes on the base of the machine sucks the air out of the bottom of the sheet thus having the plastic take the shape of the mold. This process is used to make a wide variety of products like packaging or other types of casings. Figure 4.3(b) shows a module with a clear plastic vacuumed formed shell used to protect the electronics.

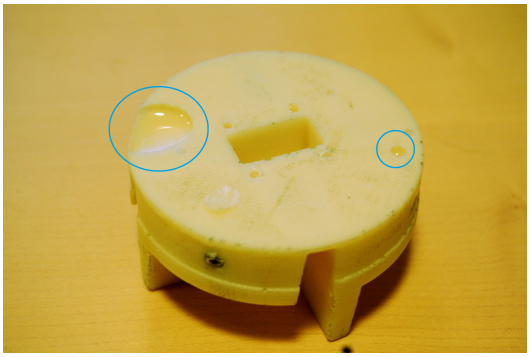
4.5 Casting

The technique used consists of pouring a liquid through a two part mold cavity. After allowing the liquid to cure, the mold can be opened and the part retrieved. While other implementations of casting exist (e.g. the disappearing wax), overall this is a very old technique for reproducing parts. The molds used in the develop-



(a) Milled connector produced in China

ment of Fable, were silicon molds based on 3D printed parts. The molded parts were made using a resin based on isocyanate, a highly toxic ingredient that when combined allows the resin to solidify. Casting is a challenging method, since it is not always clear how to design a good mold. Figure 4.5 shows a connector that was not produced properly, the blue circles mark the air bubbles that where produced during casting. We quickly abandoned this method due to the toxic nature of the raw materials.



(a) Back side of a molded connector

4.6 Injection Molding

Injection molding consists of producing parts by injecting material into a mold. Injection molding can be performed with a host of materials ranging from metals to polymers, but it is most commonly used with thermoplastic and thermosetting polymers. During the manufacturing process, material is forced into a mold cavity, where it cools and hardens to the configuration of the cavity. Once the part has hardened, ejector pins in the machinery remove the part from the mold, leaving the mold ready to produce another copy. Injection molding is a manufacturing process that is suited for mass production.

4.7 Implementation of rapid prototyping technology

The use and experimentation with various technologies has helped us to determine which ones are better suited for earlier stages while others where better at fulfilling our requirements in later stages. FDM printers helped us develop the early iterations of the connector design. In the favor of low production cost, during the early development, we began to iterate with low quality print jobs (e.g. 0.2-0.125

mm resolution in all axes). Though the cost was low, the success rate of using such printers was also relatively low, and the production time was relatively long for iterating larger parts (e.g. 18-24 hrs per module shell), so. It became clear that we had to find alternative ways to produce and further develop modules.

Laser cutting delivered production times that where much faster than the ones expected with FDM printers. This technology helped us built prototypes from the very early stages, as shown in Fig. 4.3. The ease of use of the technology and the short production time were some of the features that helped us jumpstart the first generations of modules. Vacuum forming helped cover, and protect, the electronics mounted on each laser cut frame. Using a vacuum former is intuitive and easy enough that most people would be able to quickly understand and operate the machine, however it is not clear how the mold for the vacuumed parts should be designed in order to achieve a successful reproduction of the desired part.

At a later stage we decided to try and cast connectors, since an average FDM print job for a connector was 6 hours, and the number of connectors needed was above 50 units, so it was not possible to 3D print them. We found it quite difficult to successfully cast a connector, as mentioned earlier, most of our attempts where flooded with air bubbles. The toxicity of the materials used, and our lack of experience in casting where strong enough reasons to motivate us towards finding an alternative solution. Later during the project we received a grant that made it feasible to reproduce connectors using a CNC. However, since manufacturing parts in Denmark is expensive we ended up placing the order in China.

During the last stages of development it was quicker for us to produce prototypes by 3D printing using SLS technology. Though the production costs are relatively high, it is a good compromise considering that there is no minimum lot to produce, as it was the case with the CNCed connectors. Since these costs are too high to be considered an alternative to mass production, we are currently in the last step of scaling up production we need to fine tune our current design to be able to produce it through injection molding, a good candidate for this step is Proto Labs.

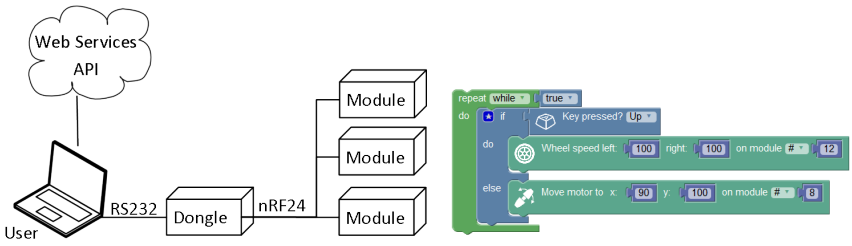
Chapter 5

Implementation of Fable

This chapter presents the design implementation and reasons behind the decision making of some of the features, many of which are not shown in detail in any of the previous publications. The chapter will first go through the system architecture, where it explains how the system works and how users can interact with Fable. Afterwards there will be a description of the system's dongle, how it works and some of the key components used for it. This chapter will also give an overview of active and passive modules, describing their key features and functions. Three active modules were designed during this project: Joint Module, Wheel Module and Sensor Module. This chapter will closes with a brief description on the cost requirements, an estimate on how much it costs to produce each of the modules as well as where it is possible to reduce costs and enable us to develop a mass producible robot at a competitive price.

The purpose of this project is to develop a system to be used in Danish classrooms and help schools teach topics previously considered difficult to learn (e.g. math, physics, chemistry). We believe that given the right platform, anyone can be a robot designer and given the right tools and environment, any child can learn and assimilate complex topics. Therefore it is necessary to deliver a system that is easy to use for non-experienced users and have a vast amount of application possibilities, thus allowing kids to exploit their creativity while learning difficult topics.

Judging from the requirements stated in section 3.3, I believe that schools are environments where modular robots could potentially adapt better than monolithic robots. In this chapter we will go through the various aspects of the system and the last section explains how our design decisions have helped us fulfill some of the requirements specified in section 3.3.



(a) System Architecture (b) Graphical Programming Interface

Figure 5.1: (a) Depicts the systems connectivity: There is a PC that communicates with a dongle through USB and the Dongle communicates with the active modules through radio frequency. (b) Represents an example code done with the graphical user interface. If the Up key is pressed the code sends a command to a Wheel Module with ID 12 to move the left and right wheels at full speed. If the Up key is not pressed then a command is sent to a Joint Module with ID 8 to move motor X to position 90 and motor Y to position 100.

5.1 System Architecture

A computer is able to communicate with the system through wireless master/slave communication. To get started users must insert a dongle to the PC's USB port. Users will then use of Fable's ecosystem of modules to start building their desired robot. Fable's active modules contain all of the electronics and power needed to perform a specific task, which could be actuation or sensing. Passive modules on the other hand contain no electronic components and serve as structural support and help give shape to the robot. Users must pair each active module with the dongle by pressing a button, either on the dongle or on the modules, to choose on which bandwidth the communication should be established. The pairing is done through color matching, where each of the 6 available bandwidth is color coded. After pairing the modules, users can run programs that send commands and requests to the modules. Through this architecture there is no need to cross-compile code and download it to each module, thus allowing users to quickly test ideas and receive immediate feedback from Fable, as shown in Fig. 5.1.

The system relies on wireless communication as a means to transfer commands from the PC to the active modules. The communication is based on the low cost energy efficient nRF24L01+ chip by Nordic Semiconductor. This radio chip allows us to have master/slave communication with numerous modules at a time, and to have high speed (2Mbps) and low latency communication, which is something that cannot be achieved with current standard protocols such as bluetooth. The

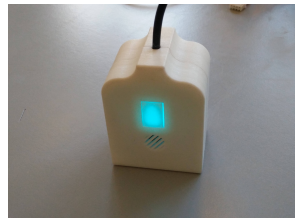


Figure 5.2: A Fable dongle communicating through the cyan radio channel.

Fable system is a heterogeneous system. This brings the advantage that users can choose and combine modules that better suit a given application. Furthermore, the system is chain based which allows users to build functional robots easier than with a lattice structure (e.g. legged robots).

Users are able to program the system using a visual programming language based on Google's Blockly [90]. The application has two programming settings: Simple and Advanced. Through these settings users can select the amount and complexity of programming blocks available in the menu. On a side panel of the GUI users can also read the IDs of the detected modules. An example code is shown on Fig. 5.1(b). Through the use of a visual programming language expert and non-experienced programmers can quickly develop their own applications. Programming the system through a visual programming language allows users to quickly sketch a desired program.

Each of the commands used to control Fable are sent to the Dongle through a USB connection. There is minimum delay of 2 ms lag that is inherit delay on the USB protocol. A ping to the Dongle has an average lag of 2.12 ms. Pinging a module has a total lag of 8.77 ms, this lag is considerably larger than the previous command, due to the sending of wireless packages to modules. More complex commands tend to take longer than a simple ping, a move motor and set speed command can take between 3.24 and 12.4 ms, depending if the request for acknowledgment is enabled or not, see Table 5.1. This means that if a 12 DOF hexapod robot is built it could run at a closed loop feedback rate of 6 Hz this includes 6 legs using the setModuleMotorPositionAndSpeed command with an acknowledgement (74.4 ms) and 6 feet sensors sending feedback with acknowledgements (74.4 ms). This performance can be improved by allowing modules to store a state vector that would include the latest sample of each sensor and send it upon request. In the current implementation modules must calculate and measure their sensory data in response to each request that has been sent.

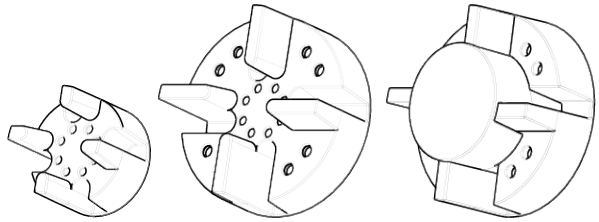
Command	Lag (ms)	Lost Packages (%)
pingDongle	2.12	0.0
pingModule	8.77	0.0
setModuleMotorPositionAndSpeed (nack)	3.24	0.0
setModuleMotorPositionAndSpeed (ack)	12.4	0.0

Table 5.1: Communication lag for selected commands in the Fable system. The values presented are the mean values taken from 100 measurements. Requesting a command with acknowledgement from a module will add 8-9 ms to the lag. There is minimum lag of 2 ms per command that is added by the USB port configuration.

5.2 Dongle

The Dongle, shown in Fig. 5.2, is used as a means of enabling the PC to communicate with the active modules in the system and these are equipped with a Nordic radio chip nrf24L01+. This was found as a better option than bluetooth since bluetooth is meant to be used as a master slave communication protocol with a limited number of links. Using bluetooth would not work, since the Fable construction system requires to have a large number of connected modules, plus it is also necessary to broadcast messages to all modules and we wanted to avoid the hassle of pairing modules as the bluetooth protocol requires it. Since there isn't a predefined protocol stack in the Nordic radio we have the freedom to define our own and avoid unwanted features from other protocols, like frequency hopping. However, in the latest version, we did implement a bluetooth chip into the design but only as a means to interface our system with tablets and smart devices and not as a communication between modules.

We intend to have Fable as a system that is maker and hacker friendly by using technology that is widely used within those communities. Since the most popular hardware community within the maker movement is the Arduino community, we decided to use the Atmega328P loaded with the Arduino bootloader and have all of the embedded code as Arduino compatible. The Dongle can be easily connected to a PC through the use of its built in FTDI to USB cable. Furthermore, the Dongle has a button with an RGB light that allows users to change the communication channel, this feature allows Fable to function in a classroom setup where it's likely that several groups would work with their own set of modules. This feature also serves as a way to prevent work-groups from interfering with each other. Each Dongle also contains a buzzer that users can easily access to emit sounds.



(a) Small size (b) Medium size (c) Compatibility of two connector sizes

Figure 5.3: Connector design: With the current design any size is possible and compatible with the rest, having only as a lower limit the small size connector diameter.

5.3 Modules

Connector

In modular robots the only physical interface between modules is given by connectors. They are responsible for maintaining a given configuration, as well as for allowing it to change. To have a construction system that would be able to adapt well to time restricted tasks we need a system with a strong connection, meaning that it wouldn't disassemble unintentionally and yet easy enough for children to build with it. The connector should be robust to wear and tear, preferably without any moving parts or mechanisms that could eventually break. Furthermore, a genderless design was preferred in order not to constraint users creativity and allow them to plug together any two connectors. For this same reason it was desired to have a system that would allow multiple connection configurations, and allow users to plug in things in X degree intervals. Lastly, we desired to have a system that was able to have modules in various sizes, meaning that users would be able to plug in small sized modules with large ones.

The designed connection system is shown on Fig. 5.3. The design has a radial symmetry that allows a pattern to repeat at 90 ° intervals. This feature allows the user to plug two modules in 4 different orientations, plus it also allows the system to maintain its compatibility between various module sizes, as depicted by Fig. 5.3(c). A set of four flanges on each connector prevents the system to detach through twisting or bending, thus allowing users to pull two modules apart as the only means to break the magnetic link.

Each connector contains a circle of magnets, where each circle is conformed by 4 pairs of magnets with alternating S-N poles. An extra ring of magnets is placed on the medium sized connector to ensure compatibility between both sizes, as seen on

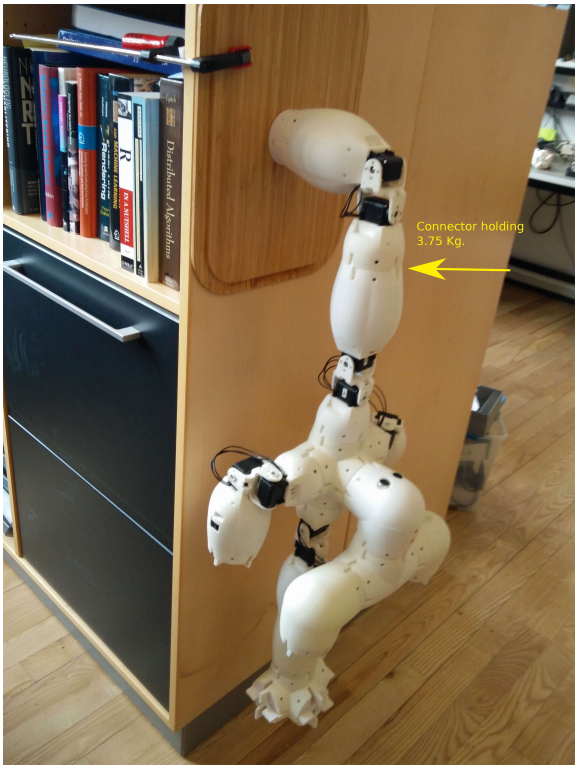


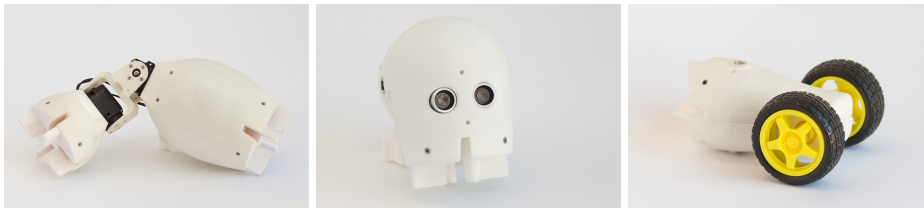
Figure 5.4: A connector can lift up to 11 modules without disconnecting, an equivalent of 3.75 Kg.

Figure 5.3(c).

With the current connector design it is quick and easy to build complex structures, and a round of tests performed at the Playware Lab show that it is possible to build a quadruped robot composed of 9 modules in less than a minute. With the current connector design one module can lift 11 other modules weighing a total of 3.75 Kg. without disconnecting, as shown in Fig. 5.4.

Active Modules

All active modules, shown in Fig. 5.5, are self-contained and contain an Atmega328P, with the Arduino bootloader, a Nordic radio chip to communicate with the dongle, a user interface that allows users to pair modules, an on board LiPo battery and charging port.



(a) Joint Module (b) Sensor Module (c) Wheel Module

Figure 5.5: Active Modules

Joint Module

Joint modules are used to enable locomotion and movement in order to interact with the environment. Joint Modules have two degrees of freedom (DOF), the amount of DOF's was fixed to two since it's a good compromise between having a bulky and expensive 3 DOF module and a limited joint with only 1 DOF. This module has currently been through four iterations that include various types of housings, some of which where vacuum formed, laser cut, molded and 3D printed, see Fig. 5.6.

The joint module electronics were designed to be compatible with the Dynaixel AX-12A motors. These motors were chosen because they have unique features which include, a communication bus that allows several motors to be daisy chained, an internal PID for position control, position feedback, torque control and a stall torque of 1.5 Nm. Dynamixel motors are robust and reliable so it is easy for users to build reliable and sturdy robots. Joint modules have a tristate buffer to enable communication between the microcontroller and the motors, since there is a need to toggle between the full duplex communication of the Atmega328P and the half duplex communication of the Dynamixel motors. Both the board and the motors are powered by a 3 cell battery 1000 mAh with an average runtime of 90 minutes per charge.

Figure 5.6 shows the three implementations of the mechanical design that were done for the system's first, second and third iterations. The first iteration made was based on an FDM 3D printer and a laser cutter, see Fig. 5.6(a). Each print job took close to 12 hours, the shapes that we were able to print were limited and the success rate was very low. Since the resolution of the print jobs wasn't very high, 0.2 mm in the XY plane, it made it difficult to design a shell that could comfortably fit the electronics for easy assembly. By using this method we where able to produce four joint modules.

The second iteration, see Fig. 5.6(b), was also based on laser cut frames but

(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

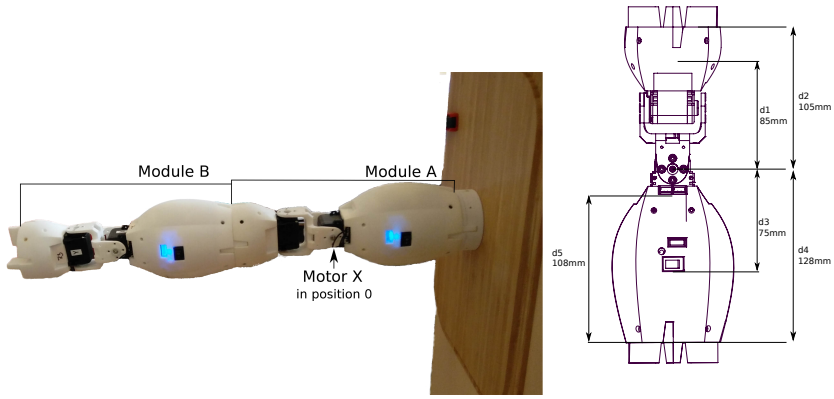
Figure 5.6: Three iterations of the Joint Module

arranged in a way that allowed us to produce modules faster than before. We built a laser cut frame where we attached the electronics and had a vacuum formed shell cover the frame. We also attempted to use silicon molds to more accurately produce shells for the joints. Molding was dropped after several attempts due to the toxicity of the materials and the safety concerns of using parts that could potentially contain a known carcinogen called isocyanide.

The laser cut frames made it easier to assemble modules, however the accuracy of the laser cutters that were available varied substantially. This meant that each time something was laser cut there was a redesigning step to make things fit properly. With this approach we where able to produce around 12 modules.

The third approach, see Fig. 5.6(c) involves selective laser synthering, a type of 3D printing that allowed us to produce very accurate shells that comfortably fit the electronics and battery of the modules. The print jobs were sent to Shapeways where each shell was produced for close to 500 DKK. This method allowed us to scale up the production to close to 30 units. When using 3D printing services it is important to reduce the amount of volume for each part as this will considerably reduce production costs. This means that instead of 3D printing a solid cube, it is better to print a hollow cube with a small opening to drain the unused material trapped inside the shape.

In this last iteration we were able to measure the stall torque of a Joint Module. According to ROBOTIS the stall torque on a Dynamixel AX12-A motors is of 1.5 Nm. If we consider Module A in the position shown in Fig. 5.7(a) but without Module



(a) Test Setup (b) Module dimensions

Figure 5.7: Joint Module lifting an identical module, with an average weight of 480 g. (b) Shows the relevant dimensions of a joint module. Distance d1 and d3 show the distance of the COG in relation to motor X

B attached, then the weight that acts on the motor is the weight of the upper half of the module with a measured mass of 180 g and with a lever of 85 mm, see Fig. 5.7(b) for a reference on the module's COG with respect to motor X. With the use of the moment arm formula:

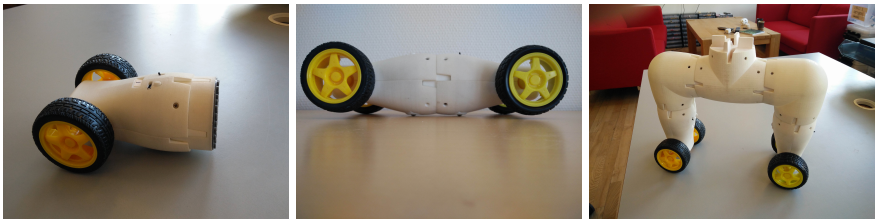
$$\tau = r \times F \tag{5.1}$$

Where τ is the moment arm, r is the lever and F is the force. We then obtain a torque of 0.15 Nm on motor X. If we have a Joint Module lift an identical robot, as shown in Fig. 5.7(a), the total torque perceived by motor X is 0.65 Nm by adding yet another module we increase the torque to 2.25 Nm, thus exceeding the limit by 0.75 Nm. In a future design the lifting power can be increased by reducing the value of d1 and d3 of the module.

Wheel Module

This module consists of two wheels, it is meant to allow users to build vehicles as well or other more complex mobile robots. The electronics from the wheel module were developed after various iterations of the Joint Module and the Dongle, which enabled us to achieve a more integrated mechanical and electronic design.

The Wheel Module is equipped with a Pololu motor driver to interface with low cost DC motors and wheels. This driver allows the module to drive forward, backward, turn clockwise and counterclockwise. The layout was done on a square shaped PCB of 45 mm, and mounting holes that allow it to fit comfortably behind



(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

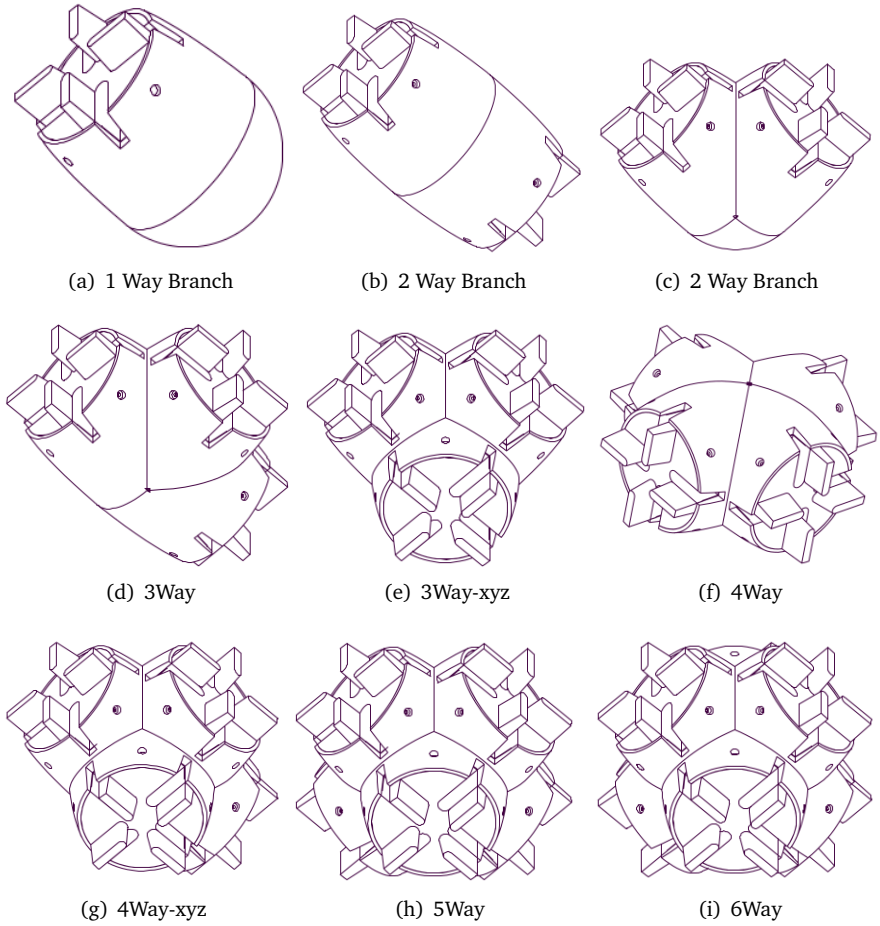
Figure 5.8: (a) Shows a wheel module with a LEGO converter. (b) Shows the problem observed when two wheel modules are connected to each other. (c) Shows an alternative solution towards building a car using wheel modules.

a medium sized connector. The Wheel Module mechanics were produced using SLS 3D printer. Each wheel motor has a stall torque of 0.078 Nm. After testing the Wheel Module prototypes we found some key aspects in the design that should be improved. One aspect is that a single wheel module has the capability of transferring robots in simple configurations like the one shown in Fig. 5.8(a). If users add a bigger module in this configuration the center of mass of the robot will shift forward and it will lose its balance, making it impossible for the current wheels to touch the ground, see Fig. 5.8(b). More complex configurations are possible as long as they can fit within the motors torque limit, see Fig. 5.8(c), however the motor torque is limited and in future designs we plan to change the actuator with a more powerful one that will also contain encoder feedback for each wheel.

Sensor Module

A sensor module allows the system to measure features of the environment it's interacting with. Currently we have developed one type of Sensor module, but the system will eventually contain several types to be able to detect gas particles, colors, sounds, light, amongst other things.

The Sensor Module, shown in Fig. 5.5(b), has an I2C interface for an IMU module, a plug for an Ultrasonic Range Finder and a generic ADC output and a user interface board. The IMU helps bring many possibilities to the Fable system such as a sense of orientation which, together with a distance sensor, can help users build robots that can react to an environment. Due to a lack of power supplied by the module's internal DC boost the ultrasonic range finder can only measure distances of up to 0.5 meters, by fixing the DC boost design it will be able to measure up to 2 m. The Sensor Module mechanics were produced using an SLS 3D printer.



(a) 1 Way Branch (b) 2 Way Branch (c) 2 Way Branch

(d) 3Way (e) 3Way-xyz (f) 4Way

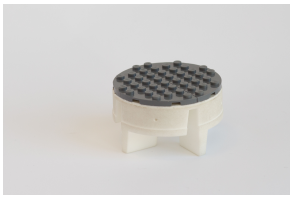
(g) 4Way-xyz (h) 5Way (i) 6Way

Figure 5.9: Passive Modules

Passive Modules

Passive modules consist of a variety of shapes made out of empty plastic shells and connectors. These passive modules help give the robot structure and shape. Passive modules are based on a cubic lattice, thus users are able to build any configuration of robots without having unused connectors thereby reducing the dead weight of the robot. The full library of passive modules can be seen in Fig. 5.9. The first passive modules were produced using an FDM 3D printer, but due to the low success rate they were later produced using an SLS 3D printer. Furthermore passive modules help us build accessories that can be used to give better functionality to a robot. The first passive module built as an accessory was

(a) Foot Module



(b) Brick Module

(c) Stand Module

Figure 5.10: Passive Modules can help give extra features and functionality to robotic constructions.

the foot module, since helps walking robots gain friction while walking, see Fig. 5.10(a). Another helpful passive module has been the Brick Module, that helps users build their own accessories by using LEGO bricks, see Fig. 5.10(b). The last example is a passive module that serves as a stand while building/testing robots, see Fig. 5.10(c). Figure 5.11 shows examples of what can be built using Fable's current library of modules[1].

5.4 Production Cost

During the closing stage of the project we sent two of our electronic designs for a quote to two manufacturing facilities, since one of the project's purposes is to have a price competitive system that schools can buy, see Section 3.3. The designs that were used were the Sensor Module and the Dongle PCB designs. The first one was sent to Seeedstudio and the second one was sent to ITEAD studio. The quotes received, shown on Table 5.2, includes all of the components of the main PCB, in

[1]The gripper module from the centaur like robot, shown in Fig. 5.11, has not been developed.

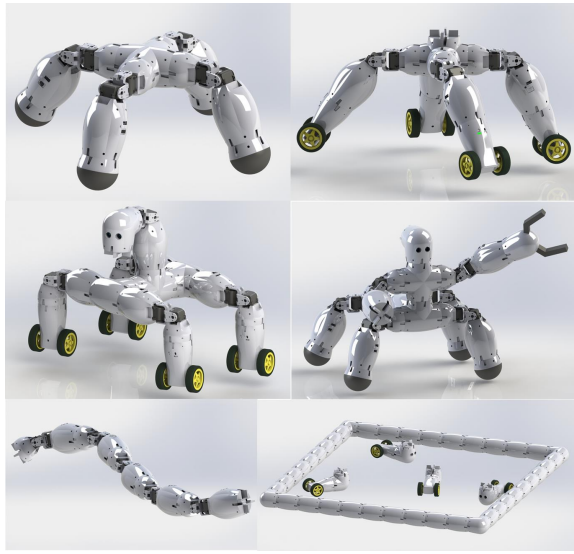


Figure 5.11: A variety of robot assemblies with Fable's library of modules.

the case of the Sensor Module it excludes the cost of the user interface board, the ultrasonic range finder and the IMU.

The first quote that we requested was for the design was the Sensor Module PCB. Through a quick view of the bill of materials I was able to determine that the production cost could be reduced by carefully picking cheaper components and excluding others that do not play a fundamental role in the module's performance (e.g. headers, sockets). With such minor changes we can expect to produce a batch of 1,000 units for at least 80 DKK per unit.

When the Dongle electronics were sent to ITEAD Studio for a quote I made sure that no headers or sockets were used and I redesigned the PCB so there would be no need of producing an extra PCB for a user interface board. Even though the Dongle's design is more complex and includes more expensive components, (e.g. BLE), due to these changes it still has a similar production cost than the Sensor Module.

Regarding the production of mechanics it is necessary to make an investment in molds. We estimate, based on quotes from ProtoLabs, that the average mold will cost around 30,000 DKK. This means that to produce a Joint Module we would require an initial investment of 150,000 DKK plus a start-up fee of 3500 DKK per part and an estimated price of 15 DKK per part. Due to the high start-up cost of mass produced goods it is wise to thoroughly test the system before committing to a specific design.

Units	Sensor Module	Dongle
100	172.51 DKK	125.85 DKK
1,000	98.09 DKK	98.91 DKK

Table 5.2: PCB production costs for the Sensor Module and Dongle electronics. All prices are unit prices given by the manufacturer, the Sensor Module production cost is based on Seeedstudio's quote, the Dongle is based on ITEAD studio's quote.

5.5 System Requirements

Requirement 1: The system must have a competitive price so most schools can afford it.

Part of this requirement is addressed in section 5.4 where we demonstrate that a low scale production of the Dongle and the Sensor Module PCB's will cost less than 100 DKK to produce and a low scale production of the plastic parts will cost less than 15 DKK per piece. In Denmark schools are accustomed to buy LEGO Mindstorms Class Sets for 30,000 DKK a unit. A LEGO Mindstorms Class Set includes 10 LEGO Mindstorms Kits, this means that the unit price is of 3,000 DKK. If we consider that a Fable Kit should include the modules used during the tests done in Trekronnerskolen and Antvorskovskolen, see Chapter 6, then a Fable Kit would consist of:

- 1 Dongle.
- 1 Joint Module.
- 1 Brick Module.
- 1 Stand.

A Fable Class Set would consist of 10 Fable Kits, if we consider the low volume production estimate in Table 5.3 then a Fable Kit can be produced for 1020.68 DKK and a Class set would cost 10206.80 DKK to produce. This means that it is possible for distributors to sell a Fable Class Set for 30,000 DKK and make a profit. If we consider a LEGO Mindstorms Class Set as a guideline of what Danish schools would be able to afford then it is safe to say that Fable is a system with at an affordable price.

Component	Amount	Production Cost	Total
Connector	4	15.00 DKK	60.00 DKK
LEGO Parts	4	1.68 DKK	6.72 DKK
Magnets	32	1.88 DKK	60.00 DKK
Dongle PCB	1	98.91 DKK	98.91 DKK
Dongle Shell	3	15.00 DKK	45.00 DKK
Joint Module PCB	1	98.91 DKK	98.91 DKK
Joint Module Battery	1	35.00 DKK	35.00 DKK
Joint Module Shell	4	15.00 DKK	60.00 DKK
Dynamixel AX-12A	2	248.07 DKK	496.14 DKK
Packaging	1	60.00 DKK	60.00 DKK
		Total	1020.68 DKK

Table 5.3: Production cost estimate of a Fable Kit, a class set would consist of ten Fable Kits.

Requirement 2 & 3: The use of the system must fit within a 45 minute lecture and it has to be easy to setup and it has to be easy to setup.

We tested requirement 2 at Trekronnerskolen, see Section 6.1, where teachers used the system in a normal setup using two lecture slots of 45 minutes each. Furthermore regarding requirement 3, in Section 5.3 lab tests have shown that it is possible to build a robot composed of 9 modules in less than a minute. We also developed an easy to install software that allows students to get their laptop ready simply by executing the installer.

Requirements 4 & 5: Teachers need to have access to relevant educational material in Danish and the learning must be in line with the official learning objectives set by the Danish Ministry of Education.

I interviewed Educational Constultants Eva Petropouleas Christensen and Jacob Kiellberg from University College Sjælland and both agreed on the possibility of developing open access educational material for the Fable System and that would be available to teachers all over Denmark. They both agreed that the Fable System was a system on which it was feasible to develop relevant educational material from 2nd grade and up. All educational material developed by educational consultants is considered to be in line with the learning objectives set by the Danish Ministry of Education.

Requirement 6: It has to be easy to pack up. Teachers have to verify that all of the robot parts were returned.

The system currently relies on modules that are fairly large and easy to distinguish. A standard Fable Class Set would consist of 40 pieces that are easy to count. Furthermore the system is simple enough to pack that teachers can ask each group of students to pack their Fable Kit back into its box, thus making it easier for teachers to pack class sets.

Requirement 7: Same tool for several school grades.

Both Educational Constultants Eva Petropouleas Christensen and Jacob Kiellberg have agreed that the Fable System can be used in various school grades probably going as low as 2nd grade. The tests in Chapter 6 also show the versatility of the system when it was tested with 2nd, 5th, 6th, 8th and highschool students.

Requirement 8: Same tool for several topics.

The system was tested at Antvorskovskolen to teach innovation and at HTX to teach math, see Chapter 6, in both cases students showed signs of enthusiasm and learning. Furthermore, in Chapter 6, I show the results of a survey done in the two main Danish conferences on educational technology where teachers and educational consultants were asked to choose subjects on which they considered relevant to use Fable. Teachers were asked to fill out the survey after they had tried the system. The most relevant topics according to our survey are math, nature & technology, innovation, physics & chemistry and IT & media. For further details on the survey refer to Subsection Teachers under Section 6.1.

Chapter 6

User Tests

This chapter presents two types of user tests that were performed with Fable: educational tests and research tests. The educational tests are the ones that were performed in a classroom environment (or similar) with kids ages 8 to 16. The research tests were performed with students and researchers using the platform to help them further develop their research.

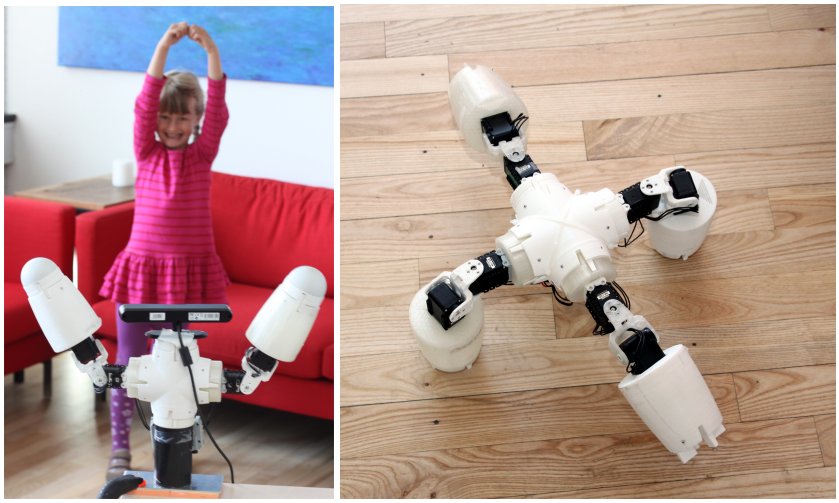
6.1 Education

This section presents a compilation of the most relevant user tests performed during the development of the system and are presented in chronological order.

Mærsk Mc-Kinney Science Center

During the early stages of development we were invited twice to test Fable at the Mærsk Mc-Kinney Science Center in Sorø, where every year there is a selection of lectures and activities that are given to the most talented kids in Denmark. In both of our visits, robotics was one of the topics given at the Science Center, and we participated by giving a 30 minute talk on Fable and proceeded afterwards to give a 2 workshops on how to program our system.

During the first year we gave a tutorial using an implementation of a RGBD camera. The setup consisted of a Fable humanoid torso, using the vision sensor as a head and two Joint Modules as arms, as seen in Fig. 6.1(a). Kids were able to program Fable and use the vision sensor to mimic the users position by mapping the users arm movements to the systems joints. A graduate student developed an API that made it easier for kids to program complex behaviours using the minimalistic programming language PicoLogo. Most kids managed to finish the given tutorial



(a) Pose Detection (b) Walking Robot

Figure 6.1: Robot configurations used in the robotics workshop at the Mærsk Mc-Kinney Science Center

and successfully implement their version of the code. However the motivation I observed was mixed, many students seemed to lack enthusiasm and some of them looked very confused (e.g. they had difficulties getting started or understanding the tutorial).

The second year we went to the Science Center we decided to try a different approach. This time the kids were supposed to program a walking robot while still using PicoLogo, see Fig. 6.1(b). We had 15 students divided into 5 groups each with their own laptop. At this development stage there was no wireless communication in our system so students had to take turns to download the code to the robot. However in this session the students were much more excited. During this implementation we did not give students a tutorial but rather gave them a quick introduction on how the robot walked and explained the few functions needed to make it walk. After this quick intro we had them freely explore the possibilities of making the robot walk. During this session none of the students managed to make the quadruped walk, but were excited to see that they were close to achieving it.

Coding Pirates

Coding Pirates is an after-school programming club for kids. The tests where performed every Tuesday evening from 17:00 to 19:00 during October and Novem-

ber 2015. At the beginning of each session kids where able to select a workshop they would like to participate in. We tested Fable with 3 to 6 kids each week. The tests consisted of seeing how kids were reacting to the system and being able to determine if the system was robust enough to be able to perform more ambitious test setups such as having multiple groups program modules while using different radio channels or ensuring that modules are unlikely to break down during extensive use.

Testing in Coding Pirates helped us develop better ways to install our software into computers, some of which included developing an installer instead of using a USB stick to copy the source files and develop a noise tolerant radio communication. The kids that attended this programming club were well accustomed to programming, thus for them it was very simple to go through a tutorial explaining most of the functionalities on each of the modules. Most of the sessions we were able to keep kids focused on solving tasks with Fable, such as using wheel modules to push away Lego bricks from a marked area, build a quadruped robot and program it to walk, and compete to build the robotic arm that could throw ping pong balls the farthest.

Overall, kids at Coding Pirates were enthusiastic and willing to play. It was better for them to program something quickly so they could start interacting and playing with each other rather than trying to solve problems in a school-like setup. Even though Fable was well received by the Coding Pirates community, I observed that we lacked exercises that would challenge kids at the right level as most of the activities where either very simple or very challenging.

Trekroner School

This tests were done at a public school located in the municipality of Roskilde in the region of Zealand. The tests were done over the course of two weeks in the month of November 2015. During the first week of testing, a team of DTU students helped us monitor the tests and also gave technical support to the teachers when necessary. The first week the system was tested each day in 3 classrooms, where each classroom was given a 90 minute lecture. Lectures where prepared for 2nd, 5th, 6th and 8th graders. Each lecture started with a teacher introducing its students to Fable. The teacher explained what was in each kit that was given to them. Each kit consisted of a Joint Module, a Fable-Lego converter, a dongle, a plate with a connector that served as a stand for their creations and a tutorial that introduced kids to programming in blockly. Once their teacher had given them a brief introduction to the kit, kids began to follow the tutorial to program Fable. When the kids finished going through the exercises they were asked to continue

with a miniproject of their own choice, which included building a catapult out of a Joint Module or building a remote controlled car with a Wheel Module.

Second graders

During our visits most second graders understood the concept of sequences of instructions but struggled to grasp concepts such as loops, conditions and reading sensors. There was a constant need to explain some topics over and over again but when I interviewed the teacher and ask her about the difficulty of programming Fable she said that it was good for kids to find it challenging to program as long as they made progress. She also commented that there was no need to simplify the graphical interface. As soon as kids where able to change the module's LED color they where jumping with joy and showing their code to other students. By the end of each lecture the room was filled with modules that were throwing ping pong balls, wheeled modules that where driving around the classroom and large structures made primarily of passive modules.

Fifth and Sixth graders

The lectures performed with 5th and 6th graders went much more smoothly, we learned that most of the kids had some experience programming, mainly through Lego Mindstorms and Scratch. It was good to see that most groups enjoyed using Fable. It was still difficult to grasp the meaning of some of the controlling tools such as while-loops and if-statements. Most of the kids showed plenty of enthusiasm and were keen on continuing programming Fable either to use the Wheel Module as a remote controlled car or the Joint Module as a catapult to throw ping pong balls at their peers.

Eighth graders

Many students had experience using block programming, some in Scratch while others using the Lego Mindstorms software. Due to this reason it was better for the eight graders to program the system using advanced features of the Blockly programming interface, (e.g. creating functions, variables). This version of the GUI is the version of blockly that unlocks the more complex blocks such as: for loops, math blocks, diy functions. Overall the eight graders programed similar things as the 6th graders, with the biggest difference being the efficiency of the program. That is the 5th graders were programming in sequential order with very little use of more advanced structures (e.g. if statements, boolean operators), while the eighth graders where using these tools more frequently.

Course	Teacher's votes
Math	68.26%
Nature/Technology	72.11%
Physics/Chemistry	54.0%
IT and Media	59.61%
Biology	24.03%
Innovation	60.57%
Languages	8.65%
Other	9.61%

Table 6.1: Percentage of teachers that voted for each of the courses. The percentages are based on the 104 participants in the survey.

Teachers

When it comes to the teachers at Trekroner there was a big difference between their level of engagement. Some were very enthusiastic about using robots during teaching, while others were more skeptical about it. Tommy one of the teachers had this to say:

"*If you come back in two years, you will probably see, I hope, that robots are an integral part of the teaching process, both for fun but also for things such as problem solving and cognitive tasks, but also to see how this could be part of, or even on its own as a separate course. Probably the hardest thing is to get it into courses that complement main subjects.*"

Tommy has a positive opinion when it comes to using robots to teach math, physics and programming. He mentions that is probably very difficult to use systems like Fable to complement other subjects, so it is possible that other teachers will find it very challenging to use robots in their lectures. In contrast, when we interviewed Bettina, a second grade teacher this is what she had to say:

"*In practice, Fable would be something that I would go back and think that, I could use it three to four times in a year. Maybe I could use it in relation to programming, it would not be something I would use in all my natural science or technical topics and let it be a part of it.*"

It is understandable why Bettina thinks that a system like Fable has limited applications when it comes to teaching second graders, after all there was no educational material showing how Fable could be used to teach other topics like Natural Science. At this stage Fable could present more of a problem for teachers to integrate into their lectures than a solution, since the system lacked teaching material and has a limited variety of modules, considering that during the tests at

Trekroner there were no Sensor Modules available. Teachers could be given support with supplementary educational material and a wider selection of modules. By interviewing teachers at Trekroner we obtained contrasting perspectives.

To get a better understanding on what teachers thought about the current capabilities of the system and its potential, we decided to do a survey and ask this question to teachers and educational consultants in Denmark. The survey took place in two events in Denmark: The Big Bang Festival and Danmarks Lærings Festival. These events are considered the most important events for educational material in Denmark. The answers obtained through the survey are shown in Fig. 6.2, there were a total of 104 replies to the question with: In which topics do you consider that Fable could be used? Where 20.1% of the teachers think that Fable is a platform that could be used to teach nature and technology, 19% say that it could be used to teach Math, 16.9% say it can de used to teach Innovation, 16.7% think it could be used to teach IT and media and 15.3% consider that it can be used to teach physics and chemistry. The least supported topics include biology with 6.7% and languages with 2.4%. The answers obtained show that teachers thoughts on using Fable to teach math, nature and technology, physics and chemistry, IT and media and innovation are similarly distributed and make up for 88% of their votes. Teachers in this survey could select all of the topics that they considered relevant, we obtained a total of 376 votes and the percentages presented are based on those votes. Furthermore Table 6.1 shows the courses and the amount of teachers that consider that Fable could be used to teach those subjects. The percentages of Table 6.1 are based on a population of 104 participants.

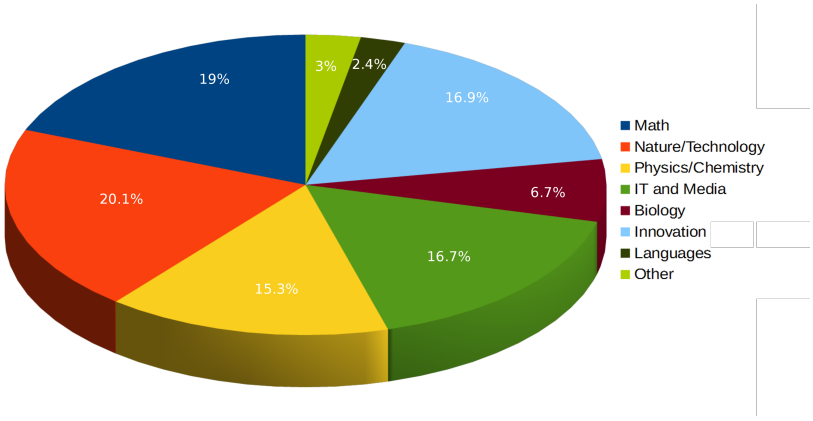
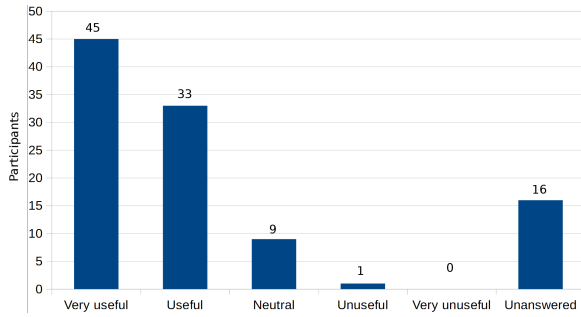
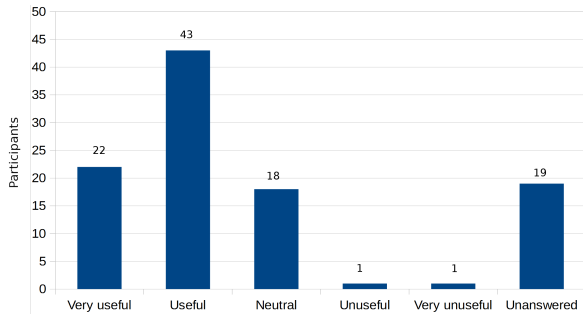


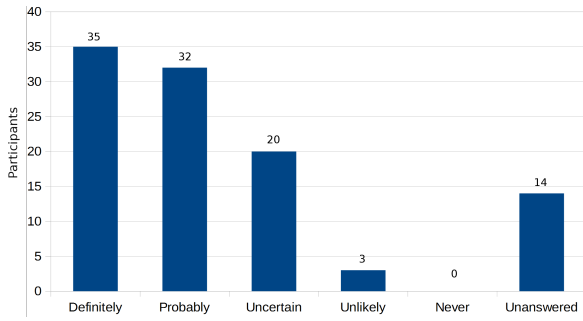
Figure 6.2: What teachers and educational consultants answered when asked: In which topics do you consider that Fable could be used?



(a) To what degree do you find Fable useful in an elementary school?



(b) Compared to the alternatives you know how useful is Fable?



(c) Would you recommend a school to buy Fable?

Figure 6.3: We asked teachers and educational consultants this questions as part of a survey that took place in Danmark's Lærings Festival and the Big Bang Festival, the two largest conferences on public education in Denmark. The survey had a total of 104 participants.

Figure 6.4: Robot setup for a math problem.

The survey also asked teachers to answer three questions with graded 5 answers ranging from very positive to very negative. The first question was the following: to what degree do you find Fable useful in a folkeskole[1]?, where 45 answered very useful, 33 said they find Fable useful, 9 answers where neutral and 16 teachers did not comment on the issue, see Fig. 6.3(a). The second question was: compared to the alternatives you know how useful is Fable? where 22 answered very useful, 43 said it was useful, 18 where neutral, 1 replied it was unuseful, 1 answered very unuseful and 19 did not reply on the topic, see Fig. 6.3(b). The last question asked was: would you recommend a school to buy Fable? where 35 replied definitely, 32 said they would probably, 20 where uncertain, 3 replied unlikely, 0 said that they would never recommend it and 14 did not answer the question, see Fig. 6.3(c). From the question regarding the usefulness of Fable within a folkeskole we can see that the system at a glance has an approval of 75% of the teachers interviewed. The second question tells us that the system has an approval of 62.5% in relation to its competitors, see Fig. 6.3(b). The third question tells us that 67% of the teachers are willing to recommend Fable to other colleagues, see 6.3(c).

Antvorskov School

During the month of January of 2016 we took Fable to Antvorskov School, where 10th grade students had to participate in an internal event called Innovation Week. During this week students were taken to a retirement home, where they talked

[1]A folkeskole is a Danish public school from 0 to 9th grade.

with the personnel and some of the elderly and learned about the daily problems that some of these people encountered. Afterwards the students went back to their school and worked, guided by their teachers, on finding solutions to some of the problems they saw during their trip to the retirement home.

Students worked continuously for three days to develop their solution, on the last day the elderly were invited to Antvorskovskolen to give feedback to the student's creations. During this period students built feeding robots, robots that would pick up a remote control for the TV, among other things. As a result of this weeks work 3 teams signed up to participate in a nation wide competition called Unge Forskere (Young Researcher), the 3 teams made it to the final round of the competition and two teams won a prize in the competition.

HTX

During the month of February 2016 we tested Fable at H.C. Ørsted Highschool where a math teacher used Fable as a tool to motivate students towards solving complex math problems. Some of the projects involved building a robot-butler, which is capable of holding a tray horizontally while it moves around, a robot that would pick up LEGO bricks and transport them and to calculate the reachability map of a robot arm. Throughout this tests students were motivated and engaged into problem solving by using trigonometry. One of the tasks that students had to solve consisted in adding a laser pointer to a Joint Module and then program the module to move the laser pointer along a predetermined path, Fig. 6.4 shows the robot setup.

6.2 Research

Social Interaction

Arnthor Magnusson was a M.Sc. student at Center for Playware during 2013. During his Master's project he worked on implementing a vision sensor for Fable. Arnthor was able to use the Robot Operating System (ROS) to interface an ASUS vision sensor, similar to Microsoft's Kinect. With the use of ROS and a vision sensor he was able to program Fable to look at a user's pose and mimic their pose by mapping the user's joint movements to Fable's. Fable was also able to recognize users and potentially adapt its behavior depending on which users are present. Arnthor's work was done during the early stages of development [4], at that time the system consisted of joint modules connected to a controller box where both the controller box and the ASUS sensor where connected to a PC running ROS.

Voice Sensing

Bjarke Heesche was a M.Sc. student at Center for Playware, he worked on developing an algorithm that was able to detect the pitch of the voice in kids and use it to control a walking robot. Bjarke worked in this project during the early stages of Fable. At that time the system lacked any sensors so his application made use of the PC's microphone and the analysis was made in Matlab. Bjarke's research opened to new possibilities for Fable like exploring how to extend the system's capabilities to detect emotions in the user's voice and adapt to those emotions [5].

Bioinspired Algorithm for Locomotion

From the 15th of January to the 15th of June I had an external stay at the Self Organizing Systems Research Group at Harvard University. During my stay I worked on the development of a bioinspired algorithm for robot locomotion. My research and work was inspired by the great adaptability that ants have, and how they are able to efficiently walk on uneven terrain, even after damaging or losing limbs. I worked towards deriving a set of control rules for a walking robot that would be evaluated on a walking robot independent from its shape. To achieve this I simulated a distributed control of Fable, by running each Joint Module on a separate thread. The result of the project was to determine a set of rules that would allow walking robots (quadrupeds, hexapods) with variable morphology to move reliably on different terrains. I began to implement the work of Righetti [91] on Central Pattern Generators with sensory feedback. On their work they present a CPG architecture that allows them to implement sensory feedback to determine whether a foot has touched the ground and adapt the CPG accordingly to help a quadruped robot walk. Unfortunately, during this period the modules were a bit unstable, since the whole system was still under development. There were bugs in the communication protocol that made it difficult to connect to several modules at once. Most of the time was spent in fine-tuning the system to a state where I could reliably communicate with many modules without any lags. Overall the work of this project helped refine many aspects of the communication and it was the first attempt towards pushing the system to communicate with many modules at a time including sensory feedback. In its current state the system is mature enough to continue developing the algorithm for a future publication.

6.3 Closing thoughts

Throughout this chapter we explored Fable as an educational platform and as a research platform and described the most relevant user tests performed during the development of the system. As seen from the teachers' quotes, the system needs to increase its versatility to become a reliable educational tool that most teachers would be willing to use. A way to win the teachers' engagement would be by increasing the diversity of modules and by delivering educational material for each school grade. We could take inspiration from the VEX Robotics website where teachers can select a school grade and download a set of recommended exercises. To improve Fable as a platform for researchers we would have to ensure that the module communication and sensor readings are fast and reliable enough to be able to do research in locomotion, swarm robotics among other topics.

Chapter 7

Final thoughts

Throughout the course of this Ph.D. we have been developing a modular robot for education with the hope that it will motivate users towards learning and socializing while building and playing with robots. As mentioned earlier, our approach uses easy to assemble units with which users can build a robot in a matter of seconds. The Fable system is unique in several ways. The system's sturdy connection is based on a connector design specifically made for user-reconfigurable modular robots. The connector contains magnets that allow a solid attachment and detachment of modules. Throughout the project we tested the connector design with kids and adults and we learned that users ages 8+ are able to attach modules quickly and with ease. The connector design is scalable allowing us to produce modules that are in a smaller scale and still be able to combine them with larger modules. This means that users can use small robotic modules to mimic finer details in a robot, such as fingers while other larger modules could be used for a leg or torso. As seen throughout this project, the connector design combines features previously unavailable on any modular robotic platform.

The Fable system encapsulates key features on each module, allows users to quickly build sophisticated robots, such as quadrupeds, hexapods, wheeled robots and swarm robots. The current system design helps introduce non experienced users to programming by using a simplified version of the graphical user interface. Once users have gained experience they are able to upgrade their blockly library by selecting the advanced version from a drop-down menu. Then when users are confident enough they can jump to standard script languages like Python or Java and start programming, plus eventually it will be compatible with research platforms like Matlab/Simulink to help academics perform state of the art research. The system's ploy glot features allow it to adapt to various user types and help users

throughout their programming journey.

Users can easily run their code on a specific set of modules by changing the radio channel used for messaging. Each of the 6 radio channels available is color coded so users can easily identify which modules RGB LED color matches the one on the dongle. This feature allows students at school to work on the same system without interfering with each others work. Users are not require to compile their code to be able to run it on Fable, enabling the user to immediately receive feedback from their creations.

The focus on Fable's design is meant to allow users to work on projects that combine a variety of disciplines usually divided by conceptual boundaries such as math, chemistry, biology and social sciences. This diversity is achieved by separating functionalities into modules. By having a simple design and a limited functionality packed into each module, we allow users to focus on being creative. When manufacturers deliberately incorporate specific lessons into toys, there is a potential to weaken a natural learning process, such as other robotic toys that hide away all complexity of their behaviour [92]. This is the reason why we consider an open ended platform a good alternative tool to aid in the learning process.

There is still much to be done in this search of how to develop a modular robot suited for education. The connector design that was presented in chapter 5 fulfills many of the requirements listed before, such as being quick and easy to connect and scalability, but there is still room for improvement. In the current design the height of the flanges are fixed, that is, it doesn't change with the diameter of the connector, where a small size would have small flanges and a large one would have larger ones. Another aspect that has to be improved is the magnetic strength in the connectors so kids to comfortably detach modules. Furthermore, the design makes use of concentric rings of magnets in order to maintain compatibility between various sizes. This increases the production cost of each connector, thus we need to redesign the connector to minimize the amount of magnets needed.

A better approach towards locking connections would be to scale flanges gradually between connector sizes, so that flanges in a connector would grow proportionally to its diameter. Thus small connectors would have small thin flanges and large connectors larger thicker flanges. There are use cases where there is a need for connectors to be permanently locked, that is so users will not be able to pull them apart. This feature is not currently supported by the system but should be implemented in future versions.

Regarding module design, all of the plastic shells have to be redesigned to fit the standards for injection molding. The electronic design for the Joint Modules and Dongle have to be prepared for mass production. The redesign not only includes

No.	Requirement	Feasible	Fulfilled
1	The system must have a competitive price so most schools can afford it.	Yes	Yes
2	The use of the system must fit within a 45 minute lecture.	Yes	Yes
3	It has to be easy to setup.	Yes	Yes
4	Teachers need to have access to relevant educational material in Danish.	Yes	No
5	The learning must be in line with the official learning objectives set by the Danish Ministry of Education.	Yes	No
6	It has to be easy to pack up. Teachers have to verify that all of the robot parts were returned.	Yes	Yes
7	Same tool for several school grades.	Yes	Yes
8	Same tool for several topics.	Yes	No

Table 7.1: List of requirements stated in Chapter 3. The column labeled Feasible states if it is possible to fulfill such requirement and the column labeled Fulfilled states if that requirement was fulfilled during the course of this project.

a finer selection of low cost surface mount components but also adds components specific to either the Dongle or the Joint Module. For the Dongle it would be wise to include a bluetooth low energy chip and have a system that would be compatible with smart devices. The Dongle lacks an FTDI chip, in contrast it uses an FTDI programming cable. Implementing the chip or a USB enabled MCU, could simplify the interface between the Dongle and the PC and may even increase communication speed. For the Joint Modules it is required to add a switch mode power converter to power the Dynamixel AX-12A motors with a 1 cell lithium polymer battery. The current implementation uses a 3 cell lithium polymer battery, making each module much more expensive. Furthermore, the motors of the Joint Modules have to be protected and enclosed so kids won't have their fingers pinched by a moving robot.

Regarding the Wheel Module it is required to improve the driving performance of the module. A way to increase the tracking could be done by redesigning the mechanics and place the wheels lower as well as by redistributing the weight of the module to increase the grip of the wheels. Other improvements to the system could be done by increasing the population of modules with a module that enables users to plug in Lego Mindstorms sensors to Fable.

7.1 Fable as an educational platform

In Chapter 3 I introduced a set of requirements that had to be fulfilled by Fable in order to consider it a system that could be offered as an educational tool for schools in Denmark. Table 7.1 displays each of these requirements.

Regarding Requirement 1 in Table 5.3 I presented an estimated production cost for a Fable Kit for 1020.68 DKK and a Fable Class Set consisting of 10 Fable Kits for approximately 10,206.80 DKK. If we make the assumption that schools would be willing to pay a similar cost for a system like Fable than what they currently pay for a LEGO Mindstorms Class Set, approximately 30,000 DKK, then this requirement is considered feasible and fulfilled. Requirements 2, 3 & 6 have been fulfilled by taking the system to Trekronnerskolen, Antvorskovskolen and HTX Technical Highschool were Fable was used within a normal educational setup. During this tests teachers were able to quickly start an activity and most students were able to setup the system without any help.

Concerning Requirements 4 & 5 I interviewed two educational consultants, Eva Petropouleas Christensen and Jacob Kiellberg, and asked them if it was feasible to develop educational material for Fable that would be alligned with the objective set by the Danish Ministry of Education. Both consultants agreed that it was feasible and showed signs of excitement and enthusiasm during their presence at the test in Trekronnerskolen. However these requirements have not been fulfilled because we have little educational material.

Regarding Requirement 7 it is considered feasible and fulfilled since we have used the system with 2nd, 5th, 6th, 8th and highschool students and all students were able to work on problems that were relevant to their school grade's learning objectives. Both educational consultants agreed that Fable is a system that could be used in several school grades.

Requirement 8 states that the same robot should be used in several topics. In order to learn about what teachers and educational consultants thought we made a survey with a total of 104 participants where we learned that most teachers and educational consultants think that Fable could be used to teach math, nature & technology, innovation, physics & chemistry and IT & media. Since Fable was not tested within the most relevant subjects, according to the survey and since it was not possible to do so in a matter that would be in accordance to Requirements 4 and 5 then this last requirement is considered unfulfilled.

If we manage to fulfill all of the requirements and manage to fix some of the technical issues of the system, then it is reasonable to assume that teachers would be able to use the platform to help them teach specific topics. In spring 2015 Fable

was granted the Proof of Concept Fund at DTU, in the hope of commercializing the system. DTU has filed a patent to protect the system. I together with David Johan Christensen, Helene Hald Christensen and Mikkel Lucas Overby have founded Shape Robotics [93] and began pre-selling Fable class sets and modules in Spring 2016 with delivery in Autumn 2016.

Paper A

Fable: Design of a Modular Robotic Playware Platform

A.1 Abstract

We are developing the Fable modular robotic system as a playware platform that will enable non-expert users to develop robots ranging from advanced robotic toys to robotic solutions to problems encountered in their daily lives. This paper presents the mechanical design of Fable: a chain-based system composed of reconfigurable heterogeneous modules with a reliable and scalable connector. Furthermore, this paper describes tests where the connector design is tested with children, and presents examples of a moving snake and a quadruped robot, as well as an interactive upper humanoid torso.

A.2 Introduction

Construction toys (e.g. LEGO) have the potential to facilitate creativity and learning by capturing the user in a mental state of focused play. In this work we explore modular playware as mechatronic construction toys that are animated and which become playfully alive as they are assembled from individual modules. To achieve this, we are developing novel interactive technology utilizing experiences from modular robotics, embodied artificial intelligence, and human-robot interactions, in order to create technological products that will motivate users to interact and play.

In this paper we describe the mechanical design of the Fable modular robotic system. The modules comprising this system can be combined by a user to create var-

ious robotic creatures, such as robotic snakes, walking robots, vehicles, humanoids, or other fantasy creatures. The long-term vision is to transform the development of robots from something done solely by experts to something so widely available, easily accessible, and motivating, that anybody is able to realize their ideas for robotic artifacts or solutions to problems encountered in their own lives.

Modular robotic systems are collections of simple robotic modules that can attach and detach from each other to form virtually endless different configurations [42, 62]. Several systems are designed to support self-reconfiguration, e.g. [94, 19], while other systems are designed for rapid robot prototyping [95], constructing walking robots [96], adaptive furniture [97], or space exploration [43].

This paper explores modular robotic playware, which are modular robotic systems designed to enable a user to construct artifacts for playful activities [98]. The modular robotic playware approach suggests that, the best way to allow users to develop robotic systems is through contextualized hands-on problem solving, which permits the users to work directly with technological building blocks in their own context. By the free manipulation of combining the technological building blocks, the user is developing technological prototypes him/herself, and these technological prototypes can be tested immediately as they are being constructed by the user in the user's context [99]. In general, the modular system becomes an object to think with, and the modularity invites the user to perform physical manipulation and reconfiguration [100].

Examples of modular robotic playware include the Topobo system which enables the user to record and playback motor sequences by programming-by-demonstration [57] and the roBlocks (now Cubelets) which utilize a programming-by-building strategy where the behavior is emergent from modules interactions with each other and their environment [61]. Further, the LEGO Mindstorms is a robotic construction kit that enables direct-user-programming of LEGO models equipped with actuation and sensing. Similar to these examples the Fable system aims to enable everyone to become a robot designer and become motivated to be creative, explore, construct, reflect, iterate, play and share.

We anticipate that the Fable system, when fully developed, will complement the previous playware examples in several novel ways: i) New robots with powerful actuation and sensing can be constructed in seconds, ii) robots can be programmed in several ways depending on the preference of the user (e.g. programming-by-building, programming-by-demonstration, programming-with-tags, and direct-user-programming), iii) socially interactive and adaptive robots can be de developed based on smart sensor modules that provide higher-level information about the user. To facilitate human-robot interaction we are inspired by robots such as Probo [101],

Huggable [102], and Kismet [103].

In the rest of this paper we describe the mechanical design (Sec. A.3) and summarize the electronic design (Sec. A.4) of the Fable system. The software architecture is described in Sec. A.5. Afterwards Sec. A.6 describes a user test with the connector design as well as tests with moving and interactive robots.

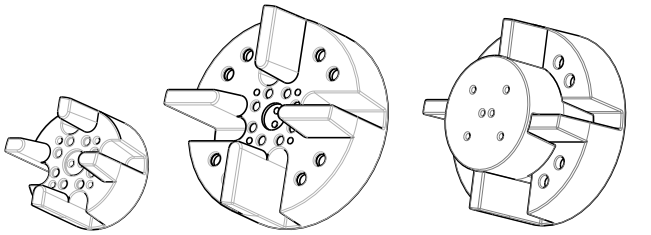
A.3 Mechanical Design

The Fable system is designed as a modular robotic playware platform, suitable for creating interactive constructions such as robotic toys. A Fable robot consists of heterogeneous modules which provide the necessary functionality to be able to perform various tasks such as sensing the user, its environment, actuating joints for movement and producing sounds. In the design of the modules we have tried to make them appropriate for interaction with non-technical users and children. Mechanical magnetic connectors allow a solid attachment and detachment between modules for rapid construction of robotic morphologies. The morphologies can be rearranged in numerous configurations based on joints, branching, and termination modules. The modules are designed to be aesthetically pleasing by utilizing rounded and organic shapes. Further, the connectors are scalable so as to allow modules of different sizes to be combined. The system is chain-based since we observe that this often simplifies the assembly of functional robots (compared to lattice based systems). The mechanical design of the Fable system is inspired by several existing modular robotic systems, such as the Cubelets [49], Topobo [57], ATRON [104], CKBot [105], and Roombots [30] and share some characteristics with these systems. This section describes the mechanical design of connectors and some different categories of modules (i.e. structuring, joint, branching, and termination).

Connectors

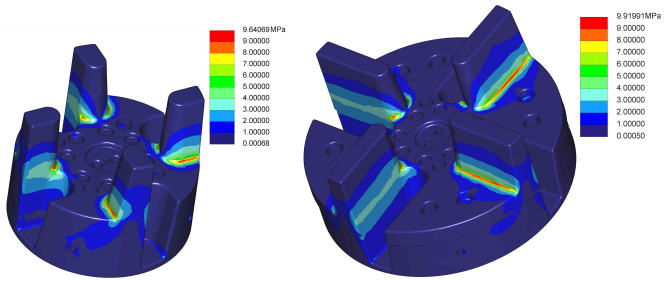
In a modular system, connectors are the only interface between neighbouring robots. They are responsible for maintaining a given configuration, as well as for allowing it to change. The following list sums up the requirements for the connector design.

- Strong connection and yet easy enough for children to disconnect.
- Robust to wear and tear.
- Transfer communication signals.



(a) Small connector (b) Medium connector (c) Compatibility of two different sized connectors

Figure A.1: Connector design: With the current design any size is possible and compatible with the rest, having only as a lower limit the small size connector diameter.



(a) Small connector (b) Medium connector

Figure A.2: Von Misses Stress analysis on a PLA design with an applied pressure of 0.45 MPa per flange on the medium size connector and 0.55 MPa per flange on the small size connector.

- Genderless.
- Multiple connection configurations.
- Scalable to maintain compatibility with larger and smaller modules.

Figures A.1(a) and A.1(b) show the overall design of two different connector sizes: small and medium. From these images it can be seen that there is a repetitive pattern every 90°, this pattern gives the user four possibilities of establishing a connection. Each of the patterns is conformed by an arch of magnets, a flange and a hole. Where each arch is conformed by a set of two magnets with alternating S-N poles. An extra ring of magnets was placed on the medium sized connector to ensure compatibility between both sizes, as seen on Figure A.1(c). The flanges mechanically lock the connection against twisting and bending and only allow detachment by pulling on the axis perpendicular to the connecting surfaces. In this

manner the magnetic force is concentrated on the same axis as the disconnection axis, this enables a strong connection between modules, requiring a force of 12 N or 30 N to disconnect the small or medium size respectively. As a result of implementing a simple design we allow users, even as young as 6 years old, to establish a connection (further details are given in Sec. A.6). The European Standard for Safety of Toys requires a toy to withstand a torque of 0.34 Nm for a ten second interval in a clockwise manner as well as in the opposite direction [106]. In order to verify this, a stress analysis was made by applying a pressure of 0.55 MPa per flange on a 56 mm^2 area on the small size connector and a pressure of 0.45 MPa per flange on a 196 mm^2 area on the medium size connector, the results obtained are displayed on Figure A.2. This implies that it is necessary to apply a torque of 2.094 Nm or 8.11 Nm in order to break the mechanical lock of the small or medium size connector respectively. Furthermore, as a tensile test the European Standard requires a dead weight of at least 90 N when the largest accessible dimension is greater than 6 mm. Both requirements are fulfilled by the design.

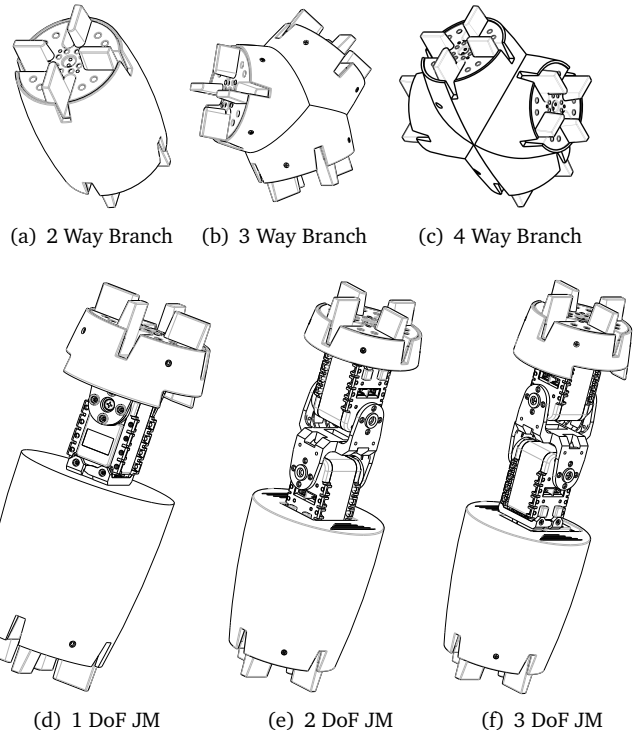
Modules

Joint modules (JM)

are robotic modules used to enable locomotion as well as interaction with the robots environment. Each articulation of the creature may have various requirements, therefore different joints have to be created. There are currently 3 different configurations of Joint modules: 1 DoF, 2 DoF and 3 DoF. The goal with all JM is to have very similar dimensions, if not identical. Figures A.3(d)-A.3(f) show the designs of the 3 types of JM. All JMs have a maximum diameter of 87 mm.

Branching modules (BM)

are units used to connect several modules together in tree-like configurations. We currently have three designs of BM. The first one being the 2 Way Branch, shown on Figure A.3(a). This module enables communication between two modules. The second one being the 3 Way Branch, shown on Figure A.3(b), enables communication between 3 modules by establishing connections at 120° intervals. The third one being the 4 Way Branch, as seen on Figure A.3(c), which enables 4 modules to communicate with each other.



(a) 2 Way Branch (b) 3 Way Branch (c) 4 Way Branch

(d) 1 DoF JM (e) 2 DoF JM (f) 3 DoF JM

Figure A.3: Fable Modules

Termination Modules (TM)

are designed to close off open connectors on a robot. TM's may add a visual expression, additional sensors, or actuators (e.g. grippers or wheels). We are currently developing several TM's, including a foot module, to enable walking creatures to walk more efficiently; a wheel module and a vision module, which makes use of Asus' Xtion PRO LIVE to equip the system with stereo vision.

Table A.1 summarizes the characteristics of the modules shown in Figure A.3.

Module	Height (mm)	Weight (g)	DoF
2 Way Branch	142	300	0
3 Way Branch	130	350	0
4 Way Branch	142	450	0
1 DoF JM	210	450	1
2 DoF JM	244	500	2
3 DoF JM	275	550	3

Table A.1: Module Characteristics, all weights given are including battery and electronics

A.4 Electronic Design

The module's electronic board has an Atmel ATMega2561 microcontroller running at 8 MHz with 256Kb of FLASH and 8Kb of RAM. The electronic boards have a connector to power and can control several daisy chained AX-12A Dynamixel servos. Further, the boards have four IR channels (half-duplex) used for neighbour-to-neighbour communication. In addition, each board can be equipped with an XBee dongle for wireless communication between modules or to a PC. All of the electronic boards have an accelerometer, a gyroscope, and the possibility to attach additional sensors and a small speaker for simple sound output. The boards are powered from a three cell 11.1V lithium-polymer rechargeable battery with a capacity of 1000 mAh. The microcontroller can be programmed externally without disassembling the modules. These custom made electronic boards are in the process of being integrated into the system. For the experiments presented in Section A.6, we used a compatible embedded system (CM-510 controller box) for centralized control of the modules. However, the software and control architecture used is fully functional and will be ported directly to electronic boards for distributed control.

A.5 Software Architecture

Ideally application programming for the Fable system should be open and accessible to non-expert users (e.g. designers, students and even kids). Thus the programming tools must abstract away many of the low level details of sensors, motors and distributed processing. Therefore the Fable software architecture has both a low-level firmware part as well as a virtual machine to provide high-level programming of robot applications.

The embedded microcontrollers in the Fable modules are running a low-level software system which has been developed primarily in C. In order to ease the development of the embedded firmware we utilize the Assemble-and-Animate framework (ASE) [107]. This gives us a high level of abstraction, a large library of components and algorithms and an asynchronous event based framework.

For inexperienced programmers the C programming language can be too complex to utilize. We have therefore chosen to provide the LOGO programming language for application development. LOGO is an educational programming language which provide a higher level of abstraction and a more natural English-like syntax than C (e.g. no braces). The Fable firmware includes a virtual machine (VM) that runs as a (non-blocking) process in the ASE scheduler and can execute LOGO applications compiled to byte-code. The VM allows the user to upload LOGO byte-

code to the modules through a serial connection or a USB flash drive. Running in a VM also means that the LOGO code can run on the modules more safely since the VM limits the program's access to the systems resources. Domain specific primitives with a higher abstraction level are included as part of the provided LOGO language. These features enable the user to more quickly develop an application and upload it to the robot.

We use a restricted version of LOGO for embedded devices, called PicoLOGO developed by Brian Silverman [108]. PicoLOGO has a reduced set of available instructions and is currently limited to only integer type variables. There is a performance penalty when executing LOGO code in the virtual machine, compared to the equivalent C. Microbenchmarks indicate a penalty factor of approximately 9.

For more advanced processing, than possible or feasible on an 8-bit mirocontroller, we utilize the Robot Operating System (ROS) [109]. ROS gives us access to a large set of software libraries such as OpenCV. The software architecture is such that we run ROS on a server which provides services (over a wired or wireless serial connection) to the LOGO code running on the modules. This architecture has for instance enabled us to develop a humanoid model, from the modules, that can speak, detect faces, and mimic postures from humans using a motion sensing input device [4].

A.6 Tests

This section describes several tests performed with the Fable system: 1) a user test performed with children to validate the usability of the connector design, 2) a snake and 3) a quadruped robot to validate the systems ability to construct mobile robots and 4) a simple humanoid upper torso to illustrate the construction of interactive robotic artefacts.

Connectors.

It was stated as a design requirement, in Section A.3, that children should be able to attach and detach the Fable modules. To investigate this requirement a user test was performed at an after school club in the Copenhagen area. At the time of testing, no modules had been prototyped. Instead the modules were represented by a solid PVC cylinders with a connector attached to one end. Both the medium and small sized connectors, prototyped with SLS technology, were tested. The diameters of the cylinders were 40 mm for the small connector and 60 mm for the medium, both of them having a height of 100 mm. The connectors had magnets attached to

them to help fasten the connection just as stated within the design description. The age group that was targeted for the design was ages 6 and up. The group tested consisted of 16 children. Their ages ranged between 6-10 years, where 9 out of the group were 6, 1 was 7, 3 were 8, 2 were 9 and 1 was 10 years old.

The children were asked to connect and disconnect the connectors several times and we observed if they encountered any difficulty. Most of the children did not have issues connecting the modules. Four kids, mainly 6 years old, had issues connecting during the first couple of tries until they figured out how to connect them. Sometimes it was sufficient for them to mimic the process, just by watching an older child succeed. We also observed that the force required for disassembly exceeded that of most children. Further tests with different connector prototypes are needed to determine an appropriate design that enables children to comfortably disconnect two modules.

Quadruped locomotion.

The 8 DoF quadruped is built by connecting a 2 DoF Joint module to each end of a 4 Way Branch. The configuration weighs a total of 2024 g, including battery and controller.

A Central Pattern Generator (CPG) based controller was developed for the quadruped (based on the CPG architecture of coupled oscillators presented in previous work [107, 110]). Four gaits were implemented in the design: forward, backward, clockwise turn and counter-clockwise turn. The current gait is controlled by the analog stick of a gamepad. The gamepad is connected to a PC which has a wireless ZigBee communication link with the robot. Only a high level control signal is sent to the robot. All CPG computation was performed locally by the embedded microcontroller.

Figure A.4: Two boys connecting a medium sized connector and a small sized connector

To study the robot's mobility the test setup was based on two types of floors with different friction coefficients. The first one being a foam based mat and the second one being a linoleum based floor. The walking speed was measured by attaching a string to the robots body and measuring the time it took it to pull the string a distance of 1.5 m. The results obtained are shown on Table A.2. The turn speed was made by averaging the time it took the robot to make 10 full revolutions. On foam, the average turn rate was 0.18 rev/s while the average value obtained on the Linoleum was 0.15 rev/s.

We observed that the quadruped moved with a reasonable speed and turning rate compared to its size and that the CPG architecture produced natural smooth transitions when switching between gaits. Further, we encountered no issues with servos being too weak or connectors disconnecting unintentionally. Pictures from a test run is shown in Fig. A.5 [1].

Snake locomotion.

This section describes the tests done with the snake configuration. Four 2 DoF JMs are arranged in series in order to achieve the morphology presented. This snake configuration weighs a total of 1771 g, including the battery and electronics.

For this configuration a side-winding gait was implemented based on the same CPG architecture as for the quadruped. The frequency of the gait was controlled using the analog stick of a gamepad.

The same types of floors were used for tests as in the quadruped test, i.e. the first one being a foam based mat and the second one being a linoleum based hallway. A side winding distance of 60 cm was used in order to determine the speed in each

[1] Videos of the different robots can be found on: http://www.youtube.com/user/centerforplayware

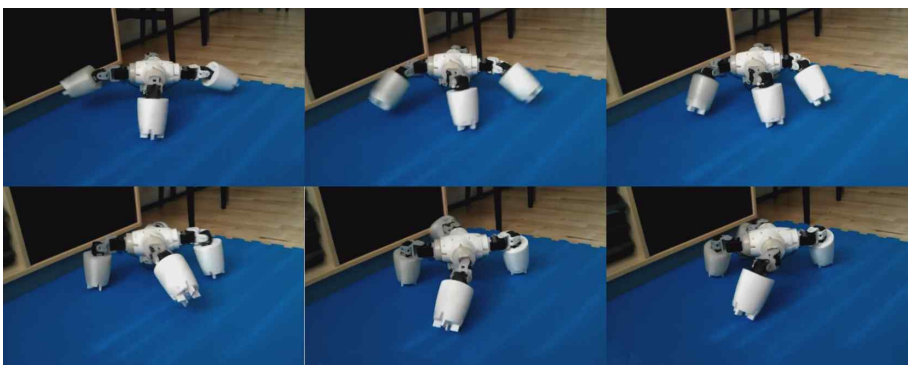


Figure A.5: Walking cycle for the quadruped configuration

Figure A.6: Sidewinding cycle for the snake configuration

scenario. It was difficult to measure more due to the fact that the mat used was relatively small and the snakes movement was not very straight and it tended to fall of the mat at longer distances. The distance was measured with two strips that marked the 60 cm. The results obtained are shown on Table A.2. Ten attempts were registered in total for each scenario. The average speed for the Foam mat scenario is of 0.17 m/s and the average speed obtained for the Linoleum is of 0.19 m/s. The results obtained in each scenario are shown on Table A.2.

We observed that although the snake velocity is slightly higher than the quadruped velocity the snake is much less controllable (not moving straight, flipping over). Moreover, we found no efficient gaits for turning or forward locomotion. To address these issues we will consider including the possibility of adding passive wheels to the Joint modules so that more effective gaits can be implemented.

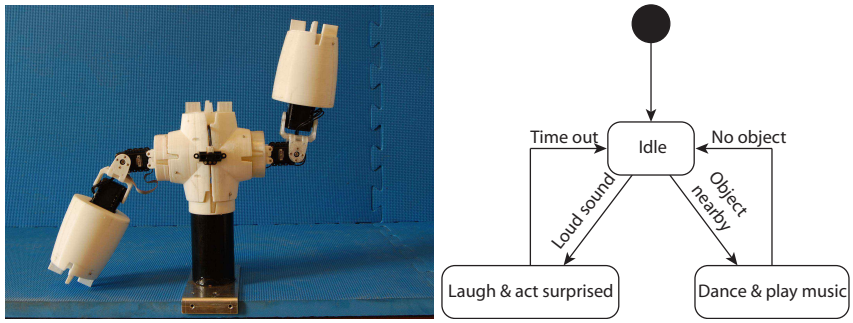
	Quadruped		Snake	
Scenario/Gaits	Forward	Turn	Forward	Turn
Foam	0.15 m/s	0.18 rev/s	0.17 m/s	–
Linoleum	0.14 m/s	0.15 rev/s	0.19 m/s	–

Table A.2: Locomotion measurements

Humanoid torso.

To study the possibility of utilizing the Fable system in interactive toy applications a simple humanoid torso was constructed. The modules used in this test were two 2 DoF Joint Modules and one 4 Way Branch. In addition an IR distance sensor is placed to sense objects in front of the robot, see Fig. A.7(a). A microphone would detect loud noises such as claps. Sound playback was performed by a PC controlled wireless from the robot.

A simple controller was developed for this robot, see the statechart in Fig. A.7(b). A loud noise would cause the robot to make a fast jerky movement and laugh with a



(a) (b)

Figure A.7: (a) Humanoid Torso and (b) statechart of its controller

baby voice to act surprised. Further, if an object was detected by the distance sensor the robot would start a dance/wave and play a children's song.

We observed that the simplicity of the robot makes it feasible for such applications to be constructed and programmed by non-expert users given the necessary programming tools. However, several additional module types at different scales are required to fully be able to construct humanoid shapes with the Fable system.

Humanoid torso with gesture detection.

Several playful applications have been developed to test the integration of the software architecture. For a humanoid torso a motion sensing camera module was mounted as a head of the robot. Applications were written in LOGO, compiled to byte-code, and ran in the LOGO virtual machine controlling the robot. The applications would use ROS' services, running on a server, to speak announcements and detect gestures made by the user. Two applications were made, one which would mirror the user's movements and one where the user should mimic the robots gestures.

The applications were tested with high school students (age 16-17) in an educational context, where the students were given exercises in programming the robot. Further, a pilot test with a young child playing with the robots has also been performed (see Fig. A.6). We observed that the system could be programmed by the high-school students with little training and that the child enjoyed herself while playing. We anticipate that further extensions and refinements of the modules and the software architecture will enable us and non-expert users to develop such custom playful applications.

Figure A.8: 6-year old girl playing with a Fable humanoid.

A.7 CONCLUSIONS

In this paper we described the mechanical design of Fable which is a chain-based and heterogeneous modular robot with a reliable and scalable connector design. The purpose of Fable as a modular robotic playware platform is to enable non-technical users to create interactive robotic constructions. We described several types of modules: joint, branching, and termination. Further, we presented usability tests of the connector design which indicated that they can be handled even by young children (ages 6 and up). The mobility of the system was demonstrated by implementing a quadruped and a snake robot controlled by a central pattern generator. Finally, the potential of realizing interactive robotic constructions was demonstrated with a simple humanoid upper torso with gesture detection.

Currently we are refining and extending the Fable modules and its software architecture based on observations from robot demonstrations and user testing. We are working on several methods to engage the user in the development and interaction with the system, e.g. by exploring different programming paradigms (e.g. tangible programming), by taking advantage of the scalable connector design, and by enabling the system to be combined with non-module elements (such a Play-Doh or LEGO bricks).

A.8 Acknowledgements

This work was performed as part of the “Modular Playware Technology” project funded by the Danish National Advanced Technology Foundation.

Paper B

Fable: Socially Interactive Modular Robot

Abstract

Modular robots have a significant potential as user-reconfigurable robotic playware, but often lack sufficient sensing for social interaction. We address this issue with the Fable modular robotic system by exploring the use of smart sensor modules that has a better ability to sense the behavior of the user. In this paper we describe the development of a smart sensor module which includes a 3D depth camera, and a server-side software architecture featuring user tracking, posture detection and a near-real-time facial recognition. Further, we describe how the Fable system with the smart sensor module has been tested in educational and playful contexts and present experiments to document its functional performance.

B.1 Introduction

Fable is a modular robotic platform under development, aimed at enabling non-expert users to design and develop socially interactive robotic creatures from various types of modules. A user can create their own Fable robots by assembling its modules into some configuration and programming it with the desired behavior based on a simple programming language. We explore the use of smart sensors modules, which are needed to achieve social interaction between a human and a robot. The smart sensor enables the robot to sense the user's behavior and respond accordantly. Smart sensors modules could for example be cameras or microphones combined with appropriate processing and mechanical encapsulation able to recognize the face or voice of the user. Mixing smart sensors and modularity

introduces a set of challenges due to high bandwidth and processing beyond what a low power microcontroller, as typically used in modular robots, can handle.

This paper presents the design and tests of a smart sensor module for the Fable platform. The module contains a 3D camera, Asus Xtion Pro Live (similar to Kinect by Microsoft) and is equipped with connectors for compatibility with other Fable modules. The smart sensor module and corresponding software architecture provides the user with a simple programming interface for user tracking, posture detection and facial recognition. Our working hypothesis is that smart sensor modules will enable non-expert users to construct socially interactive and playful Fable robots and that this will be more motivating for the user than a system with no or only primitive sensors.

In the rest of this paper we first describe related work in Sec. B.2. The Fable system is described in Sec. B.3 and the software architecture for smart sensor modules is described in Sec. B.4. Sec. B.5 describes tests performed to study the system performance and the use of the system in an educational and an playful context.

B.2 Related Work

Modular robots are comprised of independent robotic modules that can be attached and detached from one another, being able to construct various robot morphologies depending on the scenario [111]. Some modular robotic systems are able to self-reconfigure [42, 62] which allows the robots to shift their own shape by rearranging the connectivity of their parts to adapt to circumstances [37] [19], form needed tools [112] or furniture [21].

In this paper we explore user-reconfigurable robots, which allow non-expert users to physically construct their own robots from different types of modules (such as passive, actuators and sensors). This type of robots are particular suited for applications such as rapid robot prototyping [113], play [57, 60], rehabilitation [114, 115], composing music [98], and education (e.g. Mobot, Barobo, Inc. or LEGO Mindstorms). The modularity and open-endedness of these systems motivates the user to be creative, reflect, and iterate on their creations whereby they learn, train, or simply enjoy themselves.

In this paper we are considering modular robots for playful social interaction. Social interaction is being widely utilized for non-modular robots [116]. The interactive and animated iCat [117] is a type of user interface for controlling various media in a more natural way. Probo [118, 101] is a huggable robot designed to improve the living conditions of children in hospitals as a tele-interface for enter-

tainment, communication and medical assistance. MeBot [119] is a telerobot that allows people in different places to feel present and to allow for social expressions, not only from video and audio, but also expressive gestures and body pose. Playware technology [120] has also been developed as a mediator for playful social interaction for over long distances.

B.3 Fable: Modular Robotic Platform

Concept

The vision of the Fable project is to build a novel modular robotic platform, which enables non-expert users to assemble innovative robotic solutions from user-friendly building blocks. A robot is assembled by connecting two or more modules together where each module provides functionality to perform a specific task, such as sensing users, environment or actions, moving or manipulating its surroundings. With an easy-accessible software tool-kit provided, the user can create custom functionality and eventually we plan to make it easy to share their creations online with other users to try out, for inspiration, or as a starting point for their own creations. The objective is to develop this platform to enable users to realize their own innovative ideas.

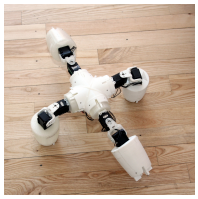
Hardware

The Fable is a heterogeneous chain-based modular robotic system which consist of various modules, such as different types of joint, branching and termination modules [3]. Joint modules are actuated robotic modules used to enable locomotion and interaction with the environment. Branching modules connects several modules together in tree-like configurations. Termination modules may add structure, a visual expression, additional sensors, or actuators (e.g. grippers or wheels).

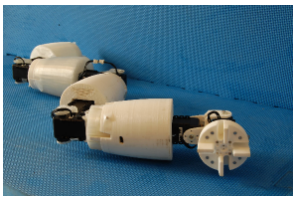
Every module, depending on its type has one or more mechanical magnetic connectors. The connectors are designed to allow rapid and solid attachment and detachment between modules. Further, the connectors are scalable to allow modules of different sizes to be combined and designed to allow neighbor-to-neighbor communication. Flanges mechanically lock the connection against twisting and bending and only allow disconnection by pulling on the axis perpendicular to the connecting surfaces.

Each module will be equipped with an onboard battery and an electronic board with an Atmel Atmega2561 microcontroller. Each board has four IR channels for communicating with neighboring modules and in addition, have the possibility to

(a) Humanoid.



(b) Four legged robot.



(c) Snake.

Figure B.1: Example configurations with the Fable system.

connect a ZigBee chip for wireless communication between modules or a PC. All modules have an onboard accelerometer and a gyroscope. The actuator modules have in addition a connector to power and can control several daisy chained AX-12A Dynamixel servos. These custom made electronic boards are in the process of being integrated into the system. Meanwhile we have used a compatible embedded system, CM-510, for centralized control of the modules.

The Fable system is still being extended with new module types to allow a wide range of different robots to be created. Fig. B.1 shows examples of humanoid, snake and walking robots constructed with the current Fable system.

Sensor Module Design

The Fable system is designed to include primitive and smart sensors:

Primitive Sensors sense simple signals, often from the environment, that can be processed by the small 8-bit microcontroller, also controlling the module, to extract meaningful low-level features (e.g. brightness, distance and temperature).

Smart Sensors sense simple or complex signals and process them to extract higher-level information about the user, the robot or its environment. Possible example includes, emotion categorization based on audio signals, gesture

recognition based on accelerations, and posture detection based on 3D depth signals.

The design of smart sensor modules should preferably be fully embedded to make the robot system self-contained and independent of external devices. This is however challenged by the limited space, processing power and battery capacity which can be embedded inside a module. Another option is to offload the sensor outputs to a server through a wireless transport to have maximum processing power, but that also introduce challenges, such as the capability to transfer large amount of data wireless, potential delays in the system, and a higher hardware complexity. At this stage of the development we have chosen to use a server-based solution with a software architecture that can be ported to an embedded device in the future. The smart sensor module presented in this paper is therefore connected using a tether to a server for processing and feature extraction. This high-level information is then offered as services to the modules wirelessly and can be accessed by the user application (as explained in Sec. B.4).

B.4 Fable Software Architecture

The section describes the software architecture of the Fable system with special focus on the server-side architecture to process smart sensor signals and provide them as services. The architecture is split into three main components: the user computer, the embedded device(s), and the server (see the diagram in Fig. B.2.).

User Computer Architecture

To enable non-expert users to program their own Fable robots, we provide the LOGO programming language for user application development. LOGO is an educational programming language coming from Seymour Papert's constructionism tradition [52] and gives a more natural English-like syntax than C (e.g. no braces). We use a restricted version of LOGO, called PicoLOGO [108], which has a reduced instruction set and limited data types. An application is created by the user, which is then compiled into a byte-code representation and uploaded through a serial connection or a USB flash drive. This architecture makes it simple for the user to program the robot and easy for us, the developers, to add new Fable specific primitives to the PicoLOGO instruction set.

Listings B.1 depicts an example LOGO program where the 3D depth camera is utilized to detect the angle of the left elbow joint of a user and maps it to a specific

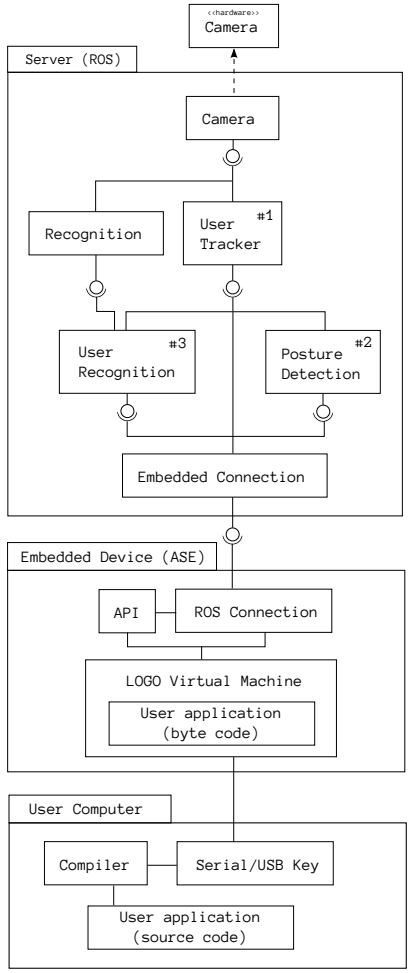


Figure B.2: Diagram of software architecture.

motor of the robot. Note that the symbol ';' starts a comment line and that the program will start on the function `onstart`.

Listing B.1: Example PicoLOGO application.

```
constants [
  [userId 1]
  [left_elbow 0]
  [motorId 1]
]

to mirror_elbow
```

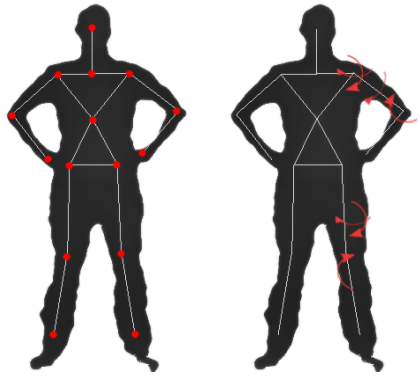
```
  ; Define a variable
  let [angle 0]

  ; Assigns the variable with elbow angle
  make "angle get_joint_angle userId left_elbow

  ; Set the motor position to angle
  set_motor_pos motorId :angle

  wait 1
end

to onstart
  ; Requests for left elbow data
  use_user_joint left_elbow

  ; Infinite loop
  forever [mirror_elbow]
end
```

Embedded Architecture

Every module is controlled by an embedded device containing a small microcontroller. The microcontroller runs the Assemble-and-Animate framework (ASE) [107]. ASE is a flexible and extendible control framework targeted for modular robots, it provides high level of abstraction, libraries of algorithms, components and an asynchronous event-driven system. A virtual machine (VM) able to execute PicoLOGO code compiled into byte-code [108] is included as a process in ASE. Hence, the user application is executed by the VM, which may call functions that use smart sensor services from the server-side. The VM abstracts many of the system resources away, such as low level details of sensors, motors and distributed processing. This enables users to more quickly develop applications and upload them to the robot.

Server Architecture

The server architecture is designed to process signals from smart sensors and offer them as services to the user application. We utilize the Robot Operating System



(a) Red dots showing joint positions available for each user.

(b) Showing different angles of arms and legs that could be calculated from the joint positions.

Figure B.3: Image showing joint positions and angles available for each user.

(ROS) [109] on the server, offering graph node structure, to create loosely coupled components which allows for task abstraction and code re-use. ROS provides a messaging framework for communication between components and also offers access to large sets of software libraries such as OpenCV and many community created components. The components role can vary from receiving sensor inputs, extract features or offering services to the user application running on the modules. The following subsections will describe some of these components and example applications.

User tracker component

The robot must be able to sense users and their movements in order to enable interaction based on body language.

As a basis for tracking and detecting users we utilize OpenNI[1] user tracker module and Asus Xtion Live Pro (similar to Kinect from Microsoft). It allows for user detection and tracking by providing coordinates for all major limbs of the body as well the angles of the joints. The component is placed on the server, as seen in Fig. B.2, component #1.

These features form the basis for other components for further processing but are also offered as a service for the user application layer. An example application

[1]OpenNI (Open Natural Interaction) is a framework created for sensor devices published under LGPL license. `http://openni.org`

we have developed with this component is a Fable robot configured as a humanoid torso with the 3D camera module on top. The robot then mimics or mirrors the users shoulders and elbow movements with its two degree of freedom joint modules (the application is an extension of Listings B.1).

Posture detection component

An efficient way for humans to communicate is verbally, but in many cases it is supported by body language, such as body posture, gestures, facial expression and eye movements. Body language can provide clues to attitude or state of mind of a person, or simply to give commands or descriptions.

To enable such a communication between a robot and the user, we implemented a posture detection system where the robot can detect pre-defined postures stored in a database. A posture is a set of labeled angles, where each angle represents a joint on the user, gathered from previously described user tracking component. The user's posture is continuously being compared to the database, detection is considered when the Euclidean distance between two sets are below a certain threshold, and then an event is triggered. This component #2 can be seen in Fig. B.2.

This functionality allows for a simple interaction between users and the Fable system. As an example we have developed another mimic application, again with the modules configured into a humanoid torso and the 3D camera module on top. In this game the robot does a certain posture with its arms and then the user mimics that posture, when the robot detects the correct posture it makes a new one, and so on.

Facial recognition component

A facial recognition service (see component #3 in Fig. B.2) is provided in order to make the robot's behavior dependent on which person it is interacting with. For example a user developing an interactive robot application might want the robot to greet and speak the name of the user's family members whenever it sees them.

The facial recognition is performed in several steps, depicted in Fig. B.4. First, the users face is extracted from an image with help from the user tracker component. Keypoints are extracted from these images with the FAST [121] algorithm, which are then given a description with the SIFT [122] algorithm. The descriptors are compared to the database with Fast library approximate nearest neighbor (FLANN) [123] matcher and on a match an event is sent. OpenCV's implementations of the algorithms were used.

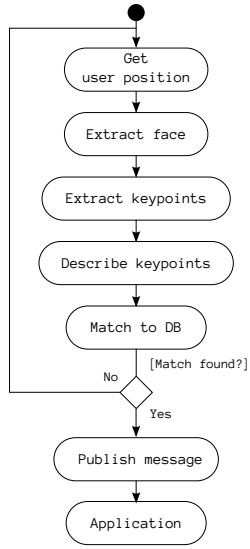


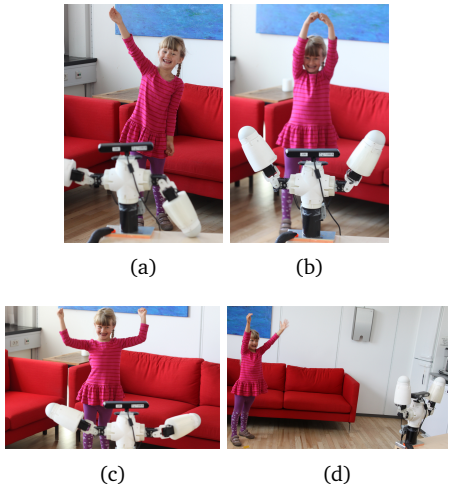
Figure B.4: Facial recognition process.

B.5 Tests

Playful User Test

The Fable system is designed to enable the construction of playful interactive robots. As a proof-of-concept, a pilot user test was performed with a 6 year old girl who interacted with a simple upper humanoid torso. The robot was pre-programmed with a LOGO application and pre-assembled from Fable modules. The robot was equipped with a smart sensor module, the 3D depth camera, directly connected to a server which was running the software architecture described in Sec. B.4. An embedded controller, the CM-510, was running the LOGO application to control the robot's actuators based on the services provided by the server. Specifically, the LOGO application would use the user tracking service to mirror the girl's movement of hers shoulders and elbows. Two special postures, identified with the posture detection service, would start and end application.

We observed the girl's reactions during the pilot test (see Fig. B.5). She was given the basic instructions about how a specific posture would trigger the mimicking. As soon as the robot started moving she reacted with positive excitement. Soon she realized the connection between the movement of her hands and the robots. As the play progressed she tried to go beyond the limits of the movements the robot can perform, for example, try to make the robot clap its hands and dance, but still appearing to be enjoying herself. In the end (after around 5 min.) she started to



(a) (b)

(c) (d)

Figure B.5: The Fable robot mimicking a 6 year old girl.

complain about her tired arms, as she had been moving them the whole time. The system was not glitch-free; there were occurrences where an unintended posture was detected, which resulted in ending the mimicking. Overall, this pilot-test confirmed that it was possible with the system to construct robotic application that would trigger play dynamics and interaction between a child and the robot. Further testing is needed to explore this in greater detail.

Educational User Test

One of the objectives of the Fable system is that it can be used as an educational tool. By tinkering with constructing robots, the user will be motivated to learn something about programming, mathematics, robotics, sensors, actuators, etc. in order to build better robots.

To test the Fable system's potential as an educational platform we have performed a user test with seven high school students. The students had little or no programming experience. They participated in a 2-hour programming exercise where they should create an application for a robot configured as a humanoid torso using the LOGO programming language. The robot should do a posture with its arms and the user is supposed to perform the same posture, then the robot does another posture and so on. This would require the participants to create functions, loops and use pre-defined functions that involve the robot moving its arms in certain positions and detecting users and their postures. A manual was supplied with the problem broken down into smaller steps and a presentation was given explaining

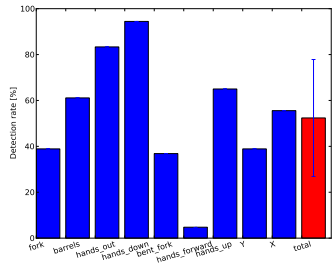


Figure B.6: Showing individual detection rates by postures, while the rightmost (red bar) shows the average detection rate.

the tools to program, compile, and upload the code to the robot.

We observed that the students quickly realized how to use the tools, while the actual programming, understanding functions and loops had a small learning curve. Most grasped the concepts quickly while others needed some extra help. By studying the manual they learned how to create functions for the different postures that the robot was supposed to perform. Some parts required extra effort, e.g. the part of detecting a user's posture. We observed that the students seemed highly motivated to understand the programming in order to make the robot do as requested. Clearly the students learned something about robotics and programming from this exercise. By letting the students define their own projects and create robots that makes sense to them, we anticipate that it will motivate the students to work concentrated for long time periods. In this process we hope that the students will learn useful skills, but the extent to which the Fable system allows for such open-ended creation and learning is a topic which we will explore more in future work.

Posture Detection

A test was performed in order to determine the detection rate of the posture detection algorithm. 9 test subjects, all male ranging from ages 25-35, were asked to perform a set of nine postures twice. No feedback was given from the system if a posture was detected or not. The camera was placed about 3 meters from the subject at 1.5 meters height.

Fig. B.6 shows the detection rates for each posture performed, while Fig. B.7 depicts the detection rate per user. The accuracy (true positives / total detection) of performed postures was 95 %, where we observed false positives being recorded while the users were moving to the final posture. Note that one posture

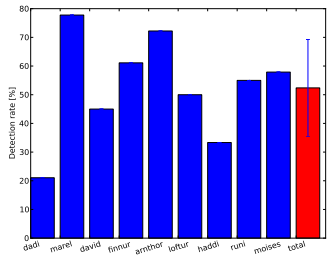


Figure B.7: Showing individual detection rates by subjects, while the rightmost (red bar) shows the average detection rate.

(`hands_forward`) had a very low detection rate. It turned out that since the hands were directed straight to the camera, it made the camera unable to correctly detect the positions of the hands. If the camera would have been placed lower this would not have affected the test. In summary the posture detection system is generally fairly capable to detect between the nine different postures, but it varies highly depending on the particular user. Further work is needed to make the posture detection component more robust.

Facial recognition

The facial recognition systems detection rate is highly dependent on the number of images in the database and specially the variation of image types, such as different illumination, face angles and distances. As the system is supposed to work in near real-time the speed of detections is a crucial factor and is therefore tested here in terms of the number of images in the database.

The test was performed on a database containing faces in different scales, distances and orientation of 5 subjects, all male ranging from the ages 25 - 35. Ten random images were incrementally added to the database at a time until 260 images had been reached, each time comparing 50 random images to the current database while recording the average time.

Fig. B.8 depicts the average time to match a face to the database with varying database size. As one can see the average time is almost constant at 0.072 s or approximately 14 faces/sec, which is sufficient for most purposes. The corresponding detection rate is 75.8 % and the accuracy is 99.3 % (true positives / (true + false positives)). We anticipate that these results are sufficient for practical applications utilizing facial recognition but this has yet to be tested.

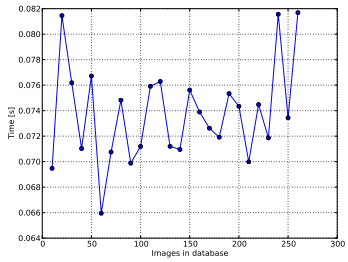


Figure B.8: Average time to match a face to database with varying database size.

B.6 Conclusion

This paper presented a smart sensor module with corresponding server-side software architecture for the Fable modular robotic system. The sensor module is based on a 3D depth camera providing user tracking, posture detection and a near-real-time facial recognition. We tested the functional performance of these services and found them sufficient for our purpose although there is still room for improvements. Further, a pilot user test was described that demonstrated a playful and interactive robot build from the system. In addition, we described a user test with high-school students who used the system to program a robot and thereby learn about programming and robotics. In conclusion we anticipate that the Fable system equipped with smart sensor modules is a step towards a new type of user-reconfigurable robotic playware that motivate its users to be creative and learn while creating socially interactive robotic applications.

In future work a more integrated/embedded version of the smart sensor module will be developed (not tethered to the server). Further, the developments of new smart sensor modules are being explored and more thoroughly tested. In addition, the Fable system is being improved to be more user-friendly and include more module types.

B.7 Acknowledgements

This work was performed as part of the “Modular Playware Technology” project funded by the Danish National Advanced Technology Foundation. Many thanks to Brian Silverman who is the developer of the PicoLOGO virtual machine and compiler used to program the Fable robot.

Paper C

Playful Interaction with Voice Sensing Modular Robots

C.1 Abstract

This paper describes a voice sensor, suitable for modular robotic systems, which estimates the energy and fundamental frequency, F_0, of the user's voice. Through a number of example applications and tests with children, we observe how the voice sensor facilitates playful interaction between children and two different robot configurations. In future work, we will investigate if such a system can motivate children to improve voice control and explore how to extend the sensor to detect emotions in the user's voice.

C.2 Introduction

We are developing the Fable modular robotic system to enable and motivate non-expert users to assemble and program their own robots from user-friendly building blocks [3]. A user assembles a robot by connecting several robotic modules together where each module provides functionality to perform a specific task, such as identifying users, characterizing the environment, moving, or manipulating its surroundings. In this paper we describe the development of a voice sensor for the Fable system that is able to measure different features (energy and F_0) of the user's voice (in particular children's). Our objectives with developing this sensor include the following:

1. Facilitate playful interaction between robots and users.

2. Enable users to create their own voice enabled robot applications.

3. Motivate users to improve the control of their voice (e.g., pitch and voice level).

In this paper, we focus on the first objective by demonstrating four applications that use the voice sensor to create games and playful interaction with the Fable system (see Fig. C.2). In future work we will address the other two objectives by integrating the voice sensor and our programming tool-kit to enable the users to also program their own applications. As well, we plan to extend the sensor functionality to detect emotion in children's voices.

Figure C.1: User testing with two Fable robots controlled using the voice sensor (voice energy and F_0 sensing). The humanoid has controllable arms and the quadruped has controllable locomotion.

C.3 Related Work

As the name implies, modular robotic systems consist of open-ended modules that can be combined in different ways to form a variety of different configurations [62], [124]. Our approach to designing modular playware [125] combines modular robotics with the concept of construction toys (such as LEGO) to facilitate playful creativity and learning. Examples of modular robotic playware include the Topobo system which enables the user to record and playback motor sequences using programming-by-demonstration [57] and the roBlocks (now Cubelets) which utilizes a programming-by-building strategy where the behavior emerges from the interaction of the modules with each other and their environment [60]. Similarly, LEGO Mindstorms, LEGO WeDo, and PicoCricket [126] are robotic construction kits that enable direct-user-programming of actuated and sensing models. In some ways, our Fable system is similar to these examples and is also influenced by Papert's learning theory of constructionism [52]. Our aim is to enable everyone to become a robot designer and encourage them to imagine, create, play, share and reflect (i.e., the kindergarten approach to learning [127]). We hypothesize that by enabling the creation of socially interactive robots, we can motivate a wider range of users to participate in this activity. In order to create these robots, we must develop sensors that can reliably interpret human communication.While, much research has been dedicated to machine recognition of human communication (e.g., [128]) and its applications to social robots (e.g., [129],[103],[101],[102]), our approach is to simplify the sensor design by limiting the detection to simple units of communication (e.g, gestures). Previous work along these lines to track and detect a user's posture using the Fable system has been promising [4]. In this paper, we limit the sensor to the detection of a single acoustic variable, the fundamental frequency or F_0, and then examine if playful interaction can be sustained using input only from this detector.

C.4 Voice Sensor

While the main focus of the sensor is on the estimation and tracking of the user's F_0 over time, we must first detect if the user is vocalizing. Thus, the voice sensor consists of two elements: an energy detector and and F_0 estimator.

Energy Detector

The simplest approach to detecting if a user is vocalizing is to measure and track in real-time the acoustic energy picked up by a microphone. A simple approach is

to calculate the root mean square (RMS) of the signal over some window. However, to reduce the number of computations, the signal is squared and lowpass filtered using an exponential moving average (EMA). Thus, the energy E, for a given signal $x[i]$ is estimated as follows:

$$y[i] = \alpha * y[i-1] + (1-\alpha) * x[i]^2 \tag{C.1}$$

where α is a weighting factor. For our application, with a sampling frequency, F_s of 8 kHz, we set $\alpha = 0.990$. Thus, the time constant of the EMA was 15 ms and resulted in a relatively stable measure of E during vocalizations. The output is energy in dB relative to $E_{ref} = 1$ is as follows:

$$E_{dB}[i] = 10 * log_{10}(y[i]) \tag{C.2}$$

If a signal is detected with an energy level, $E[i] > \beta$, where $\beta = 0.05$, it is assumed to be a sound produced by the user.

F_0 Sensor

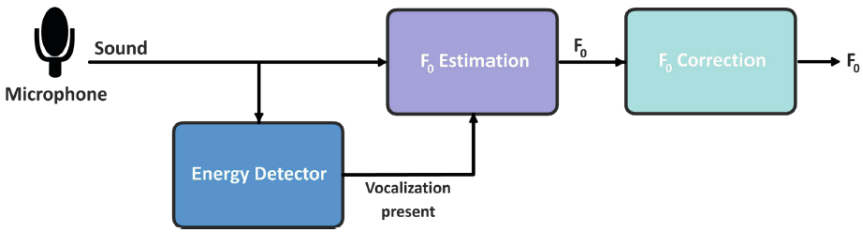


Figure C.2: Elements in the F_0 sensor.

In speech, F_0 refers to the fundamental frequency, which corresponds to the frequency at which the vocal folds open and close. However, not all speech is voiced (i.e., uses the vocal folds to produce a periodic acoustic excitation). If the vocal tract is sufficiently constricted, then the air flow will become turbulent, which produces sound. As well, for plosive sounds, airflow can be stopped and the pressure can be allowed to build significantly before the air is released in a burst. Thus, speech can be separated into voiced (periodic) or un-voiced (aperiodic) sounds. However, when estimating F_0, we currently assume that the signal is always voiced. In the future, we plan to add a harmonicity detector to determine if the vocalizations produced are voiced or unvoiced. As with the energy detector, the sensor is designed to estimate F_0 in near-realtime (i.e., low latency) to facilitate interaction. Further,

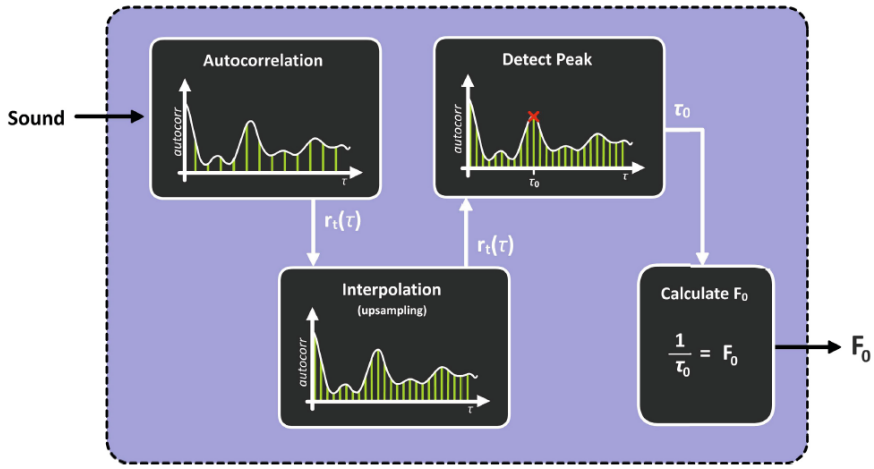


Figure C.3: Elements in F_0 estimation.

as the goal is to develop inexpensive modular sensors, the algorithm used should be feasible to implement using current embedded processors for autonomous control. The F_0 sensor consists of three major elements (see Fig. C.4). The energy detector, described previously, is used to determine if the user is vocalizing. If speech is detected, the second element estimates F_0 of the microphone signal. The final element, F_0 correction, monitors the F_0 estimates to detect and removes outliers caused by noise or unvoiced sounds.

F_0 Estimation:

The F_0 estimator uses an autocorrelation method based on [130], which is simple and computationally efficient (see Fig. C.4). The autocorrelation function $r_t(\tau)$ is computed for short time windows of lengthWauto. If the signal is periodic then the autocorrelation function will have peaks at lags τ corresponding to the period of the signal. Thus, the fundamental frequency is estimated by finding the time lag corresponding to the first peak in the autocorrelation function. To improve performance in picking the correct peak (i.e., the one corresponding to the fundamental frequency), two modifications are made.

First, the autocorrelation function is multiplied by a decay function.

$$r_t(\tau) = \sum_{(j=t+1)}^{(t+W_{auto})} [x_j x_{j+\tau}(1 - \tau/\tau_{max})] \tag{C.3}$$

Second, the search for the peak is restricted to the interval $\tau \in [\tau_{min} : \tau_{max}]$. Thus, by restricting the range of possible lags, the range of F_0 estimates is also restricted. While the average fundamental frequency for a child is 300 Hz for speech [131], F_0 can be higher for non-speech vocalizations such as humming. Thus, to cover the accessible frequency spectrum while keeping the computational costs low, we set τ_{min} to correspond with an F_{max} of 800 Hz. To allow the sensor to also track adults, τ_{max} was set to correspond with an F_{min} of 44 Hz. To further reduce computational costs, the modified correlation function was only computed over the interval $\tau \in [\tau_{min} : \tau_{max}]$. As the sampling frequency of the signal is increased, the computational costs of the autocorrelation function increase significantly. Thus, the sampling rate of the sensor, F_s, was set at 8 kHz. However, this low sampling rate resulted in poor precision in F_0 estimation, particularly at higher frequencies. Thus, the modified autocorrelation function was interpolated before searching for the maximum peak. In summary, the output from the F_0 estimator is the frequency that corresponds with the lag of the maximum peak of the modified autocorrelation function over the interval $\tau \in [\tau_{min} : \tau_{max}]$. Using sinusoidal test signals, an effective estimation range of 120–800 Hz was found. This covers the range of frequencies we would expect for children's, adult women's, and most adult men's vocalizations.

F_0 Correction: While the F_0 estimator works well, some artifacts are observed in running speech due to unvoiced speech sounds or when the sensor is placed in a noisy environment. Thus, an F_0 correction stage was developed.

For outlier detection, the F_0 estimates are monitored and the interquartile range (IQR; i.e., range between the first and third quartile) is estimated. Using this, upper and lower boundaries are set to $Q_3 + 1.5 * IQR$ and $Q_1 - 1.5 * IQR$ respectively. Estimates outside this range are considered to be outliers. Before discarding the outlier, an octave correction is tested. The outlier is shifted up or down by an octave towards the frequency range where most F_0 estimates are present. If the difference between two neighbour estimates is less than 10 Hz, the correction is considered to be successful. Otherwise, the outlier is discarded.

Performance: An example of F_0 estimation is shown in Fig. C.4. To evaluate the performance of the system, 1150 children's utterances [132] were tested. Each utterance was processed by both the voice sensor and Praat [133] and the F_0 estimates were compared. As there were occasions where Praat detected voiced speech but the sensor did not (and vice versa), the comparison was restricted to time points where both Praat and the voice sensor produced F_0 estimates.

Overall, when F_0 correction was not used in the voice sensor, the mean error was 14 Hz. When F_0 correction was employed, the mean error was reduced to 4

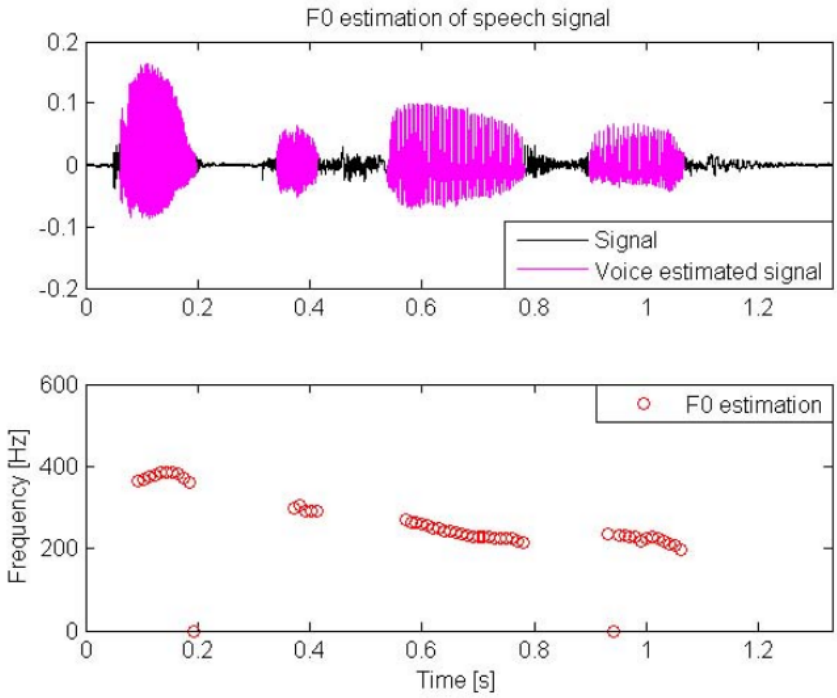


Figure C.4: Example of voice and F_0 estimation for a speech signal.

Hz. In summary, the system is sufficiently accurate for our applications, but we can improve its performance in future work.

C.5 Test of Applications

Test Setup

A total of four different applications were tested, across two configurations of the Fable system (one using a humanoid and three using a quadruped; see Fig. 1). Children's vocalizations were recorded using headsets connected to a computer. The voice sensor processing was conducted in Matlab running on a standard PC, which sent commands over a wireless serial connection to control the Fable actuators.

Four girls and four boys ranging in age from 7-10 years participated in the testing. The children were divide into four pairs according to their age: younger girls (7-8 years old), older girls (10 years old), younger boys (8-9 years old) and older boys (10 years old). As the average F_0 can vary across age and gender the individual parameters $F_{0,min}$ and $F_{0,max}$ were measured for each child prior to the

test. The aim of the tests was to demonstrate if the Fable system and the voice sensor can be successfully used by children to create playful interactions. The tests were designed as small games in order to motivate the users to engage and play with the system.

Humanoid Fable Application

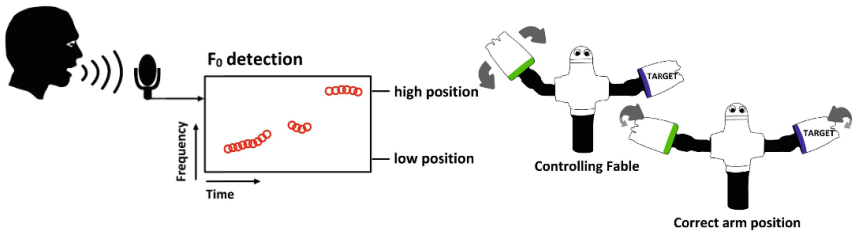


Figure C.5: F_0 control of humanoid Fable. The user controlled arm changes its position according to the F_0 produced by the user. Once the target F_0 is produced, the robot starts waving both hands.

The first application was developed to provide an introduction to how the users could control the robot with their voice. In the game, the goal was to angle both of the Fable arms the same way (see Fig. C.5). One arm, the target, was initially angled by the experimenter using a gamepad. The other arm was controlled by the F_0 of the user's voice. The user hummed or sung a tone into the microphone and the arm would raise or lower based on the F_0 that was produced. Once the controlled arm matched the target, the Fable started waving its arms. This waving continued for as long as the user could hold the tone. This task was selected as a good introduction because the children received clear visual feedback from the robot (i.e., the angle of its arm) that corresponded to the pitch they produced. Three different random target values were tried for each of the eight children.

The task was easily understood and all of the children were able to complete it. However the older children were slightly better at the task. Initially, two of the younger children had difficulties because they were changing their pitch too fast and/or in large steps. As well, the younger girls required more instruction before understanding the relationship between their voice and Fable's arm position. Regardless, the most common strategy for completing the task was to start at a random pitch and adjust it up or down seeking the target position. Even though the task was developed as an introduction to the concept of controlling a robot by

voice, the children were very engaged in the activity and appeared to be enjoying themselves.

Quadruped Fable Applications

In the following applications the objective was to move the quadruped robot through a track with four gates. The F_0 of the users' voices was used to select between four different movements (forward, backward, left-turn, right-turn). In addition to the robot's movement, visual feedback was also provided on a computer screen. The F_0 being produced was plotted as a large dot that moved over a background that illustrated the target pitch corresponding to each motion. Three different applications were tested:

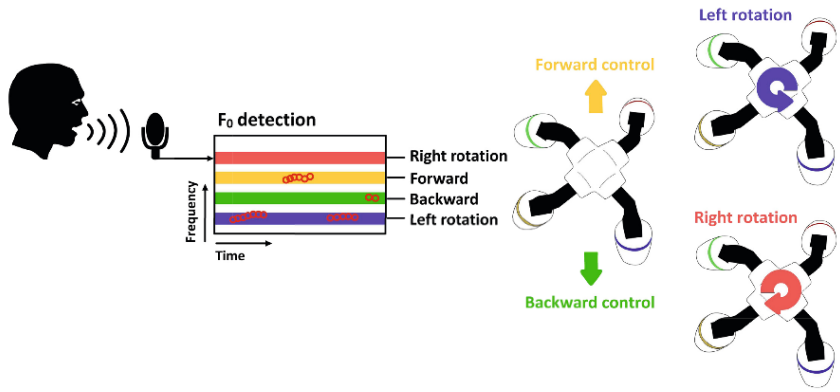


Figure C.6: F_0 control of quadruped Fable. By producing an F_0 that corresponds to one of the targets, the user can select different movements and control the robot.

Single Player - In this game, there were two different levels of difficulties. At level 1, there were only two pitch targets corresponding to forward and left-turn movements. At level 2, there were four pitch targets, each corresponding to one of the four movements (see Fig. C.5). Two Player - Collaborative: In this game, two players collaborated to control the quadruped using two pitch targets. One user controlled forward and backward movement while the other controlled turning left and right. Two Player - Competitive: In this game, two players competed to gain control over the quadruped and move it to a specific area to win the game. One user controlled forward and left-turn and the other user controlled backward and right-turn. The user that was loudest, while maintaining a target pitch, controlled the robot. However, to avoid the users screaming as loud as possible, a maximum

value was set for the voice energy. If the user exceeded this level, control was lost until the voice level was reduced below the maximum threshold.

All eight children participated in the single player game at level 1. The four oldest quickly understood the task and learned how to shift between tones in order to get the robot moving through the track. In contrast, only one of the younger boys performed well. The others needed many attempts before completing the task. The two older boys tried level 2 and were able to control the robot successfully. While we anticipated that level 2 would be more difficult, the boys were able to complete the route faster than at level 1.

As the younger children had difficulty with the singleplayer game it was decided that they would participate in the collaborative two-player game with the older children. The collaborative game added an additional complexity as the users needed to communicate with each other without triggering the robot by mistake. This challenge engaged the children to communicate with gestures or mute the microphone, and they were surprisingly good at completing the task. One of the boys commented that the single player game on level 1 was a bit hard, because he could only turn left. He also felt that level 2 was both easier and harder. It was easier because he had more control options, but it was more difficult because of all the target pitches he had to remember. However, he felt the collaborative game was just right, and really fun, because they had access to all the four actions, but only had to remember two notes at a time.

The competitive two-player game was only tested with the older boys. This application turned out to be the most difficult of the four tested. It was difficult for the boys to control the tone and the power of their voice simultaneously. As the position of the microphone significantly influences the energy measurements of the voice, one of the boys adjusted the microphone position rather than the level of his voice.

Across all the applications, the younger children had difficulties holding a specific pitch for a long period of time. Instead, they would hum a small melody. This approach might have been easier for them had they been instructed to change between two melodies instead of two tones. As a result, the older children seemed to have more fun when playing with the system. The two older boys continuously asked to play the collaborative game again after finishing. If the interactions are not changed, it is suggested that future tests focus on the older age group (i.e., 9 years and older). In summary, these example applications confirmed that we can use the voice sensor to facilitate playful interaction between the children and Fable robots. In addition, we believe that the children were also learning to better control their voices but further work is needed to test this.

C.6 Conclusion and Further Work

This paper presented a voice sensor for the Fable modular robotic system. The sensor measures the energy in the user's voice and estimates F_0, which is related to the pitch of the voice, with very low latency. The performance of the F_0 estimation was tested and found sufficient for the purpose. Four different applications utilizing the voice sensor were tested with eight children and we found that the system facilitated interaction that was challenging, playful and motivating. In future work, more voice sensor functionality will be added and the possibility of extending the current processing for use in emotion detection will be explored. In addition, we will integrate the voice sensor with our existing software tool-kit to enable users to program their own applications.

Acknowledgements. This work was performed as part of the "Modular Playware Technology" project funded by the Danish National Advanced Technology Foundation.

Paper D

Fable: A Modular Robot for Students, Makers and Researchers

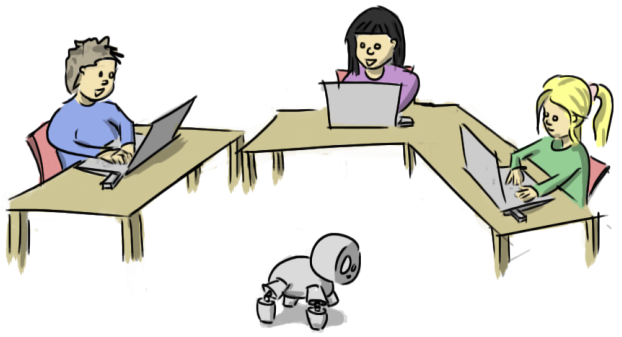
D.1 Abstract

The vision of the Fable modular robotic system is to transform the development of robots from a process performed mainly by experts, to an easily accessible and motivating activity that enables a large range of users to assemble and animate their own robotic ideas. To achieve this vision, the Fable system consists of a range of modules equipped with sensors and actuators, which users can easily assemble into a wide range of robots within seconds. The robots are user-programmable on several levels of abstraction ranging from a simple visual programming language to powerful conventional ones. This paper provides a brief overview of the concept, design and state of development for the second version of the Fable modular robotic system.

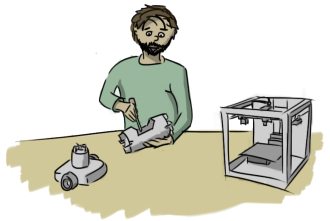
D.2 INTRODUCTION

Today's world is filled with consumer products that constantly encourage us to buy and not to build. Taught to us from an early age, plagiarism and copyright policies serve as mental barricades that dry out our curiosity, creativity and collaboration [1]. In this work we seek to revitalize and quench our users thirst for knowledge within the domain of robotics. We believe that, given the right tools, anyone can become a robot designer.

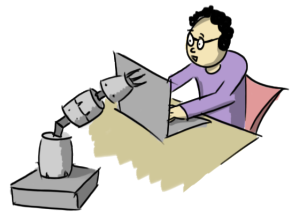
In this paper we present the concept and design of Fable, a mechatronic construction kit that allows users to playfully build and program their own robots. Our



(a) Students



(b) Maker



(c) Researcher

Figure D.1: Prospective Fable users with different interests, objectives and level of experience.

objective is to motivate users both towards making their own robots and sharing them with others, that is as a DIY (Do it Yourself) kit and as a DIT (Do it Together) kit.

We take inspiration from other modular robotic systems, which consist of a collection of simple robotic units that can attach and detach from each other to form a wide range of configurations [15, 62]. While several systems are designed to support self-reconfiguration, e.g. [94, 19, 30], Fable takes its inspiration mainly from user reconfigured and interactive systems such as Roblocks/cubelets [60], MOSS [134], Topobo [57] and LEGO Mindstorms.

The rest of this paper starts by describing the concept behind Fable (Sec. D.3). It continues by presenting the design of Fable, that is: mechanics, electronics and software (Sec. D.4). Further in Sec. D.5 the paper exemplifies how robots are assembled and programmed.

D.3 Fable Concept

We have designed Fable as a modular robotic platform with a focus on our users' needs, ranging from a classroom of kids, and after-school clubs, to hobbyists/makers and even researchers, as illustrated in Fig D.1. This diversity in usability is achieved by encapsulating key robotic functionalities into modules that can be combined in numerous configurations utilizing a shared connector and communication system. This gives us the freedom to design basic modules for kids and high-end modules for researchers while making it easy for makers to start building their own. To support this diversity of users we enable them to program the system by using their preferred programming language (Blockly, Python and Java are currently supported). The Fable system is designed to support constructionism and creative thinking. We took inspiration from Resnick's et al. design principles [135] to empower users to be innovative. Furthermore, through a rapid reconfiguration of modules we support the user's mental state of flow [136].

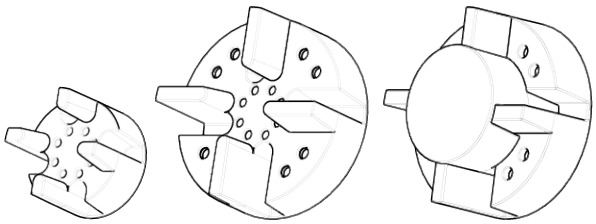
D.4 Fable Design

This sections provides an overview of the design of our second version of the Fable system, details about the first version can be found in our previous work [3, 4, 5].

Mechanics

Our approach uses powerful, yet easy to connect modules that allow users to assemble a functional robot in a matter of seconds. The Fable system is divided in active and passive modules. Active modules contain a set of electronic boards with a microcontroller, onboard power, and a radio device for wireless communication with a PC. These modules also provide functionalities through actuation and sensing, e.g. one active module design is a 2 degree of freedom joint, see Fig. D.4(a). Passive modules consist of a variety of shapes made out of empty plastic shells. These passive modules help give the robot structure and shape, e.g. a 'Y' shaped module is used to connect three modules together and an 'X' to connects four modules. Both can be seen in the robot configuration in Fig. D.4(b).

A key feature in modular systems are the connectors since they serve as the only contact surface between modules. Our current connector design, illustrated in Fig. D.2, is genderless and four way redundant, which allows our users to explore several connection possibilities between modules. Each connector has at least one ring of magnets that attaches to a matching set on the connecting end. The connector uses



(a) Small size (b) Medium size (c) Compatibility of two connector sizes

Figure D.2: Connector design: With the current design any size is possible and compatible with the rest, having only as a lower limit the small size connector diameter.

a set of flanges that lock the modules allowing only the user to disconnect them by pulling them apart. With this design we obtain a strong connection between modules and yet it's easy enough for children to disconnect. The connector design is scalable, meaning that it is compatible between different sizes, giving us the possibility of combining large modules with small ones, as illustrated on Fig. D.2(c).

Electronics

For the Fable system we have developed a set of electronic boards that combined with commercially available boards give us a modular electronic configuration. The electronic boards are designed for simplicity, low production cost, flexibility and hackability. The modular approach enables us to create different active modules by mixing electronics boards in new configurations. Table D.1 describes the electronic boards currently used in Fable. Different active modules and a radio dongle will use a specific subset of the electronics modules. As we develop new types of Fable modules we will develop new modular electronic boards to support them.

System Network Architecture

The underlying objective of the Fable network architecture is to make the robot programming as simple and flexible as possible. Due to a low lag radio communication link to the modules, the user can program the distributed robots as if it was centralized and connected directly to the PC. Therefore, the user avoids the difficulties of cross-compiling, downloading program to robot and debugging a distributed embedded platform.

The network architecture of the Fable system is shown in Fig. D.3. The user PC is serially connected to a dongle which provide a shared 2 Mbit radio communication

Board Name	Details	Description
Module Board	11.1v, 1000mAh LiPo battery	Main mother board for active modules
Dongle Board	5v USB powered	Main mother board for dongles
User Interface Board	RGB diode, button, buzzer, recharge plug, on/off switch	Functions for user feedback and direct interaction with module/dongle
Motor Board	RS232 half-duplex dual buffer	Interface board for serial control of Dynamixel motors (e.g. AX-12A)
Arduino Pro Mini	AVR Atmega328, 3.3v, 8MHz	MCU module for dongles and modules
Radio Board	NRF24L01+, 2.4GHz, 2MBit, SPI interface	Wireless communication between modules and dongles

Table D.1: Overview of electronics boards used in Fable

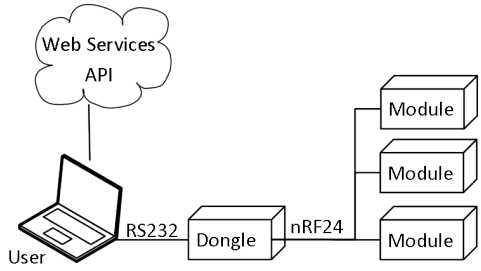
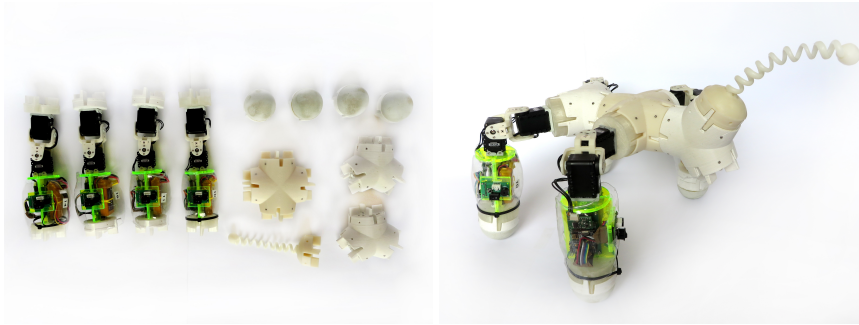


Figure D.3: Network architecture of the Fable system.

link between the user control application and the modules. Modules are addressed using an ID and their module type. Web services can be used to enhance the possibilities of the user application, currently we use a web service API for speech generation. Although the radio communication is shown here as a master-slave architecture, it can also function as a peer-to-peer network. This could be exploited in certain research studies, e.g. on distributed control. In addition, we plan to exploit asynchronous communication between different dongles to enable the different users to program collaboratively by writing different part of the program and remotely calling functions developed by other users.

User Programming

The user develops the system application from a personal computer using one of the supported programming languages. When the user executes the application



(a) Modules (b) Quadruped

Figure D.4: (a) Four active joint modules and eight passive modules. (b) A Fable quadruped robot assembled from the above twelve modules.

program it is not cross-compiled but run locally on the PC. Simple APIs enable the user-program to call functions on the remote modules through the dongle connected to the PC (based on module IDs). The round-trip lag for a simple remote procedure call is around 4.2 ms, longer if the remote module need to perform processing. This is sufficiently low for most applications and can be reduced even further in future work.

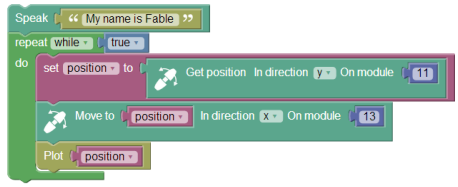
For non-programmers to quickly start developing applications a graphical user interface (GUI), based on Blockly [90] is used. Blockly is an open-source framework for building application specific visual programming languages inspired by the Scratch programming environment [137]. An example of a Blockly program for controlling a simple interactive Fable robot is shown in Fig. D.5(a). The source output generated from Blockly is written in Python. This string of Python code is send from the Blockly JavaScript web-application through an HTTP request to a Python server which executes the Python control application.

More experienced users can also write applications directly in either Python or Java. Further, we plan to add Matlab support for research and university education in future work.

D.5 Example

This example illustrates a simple typical robot that could be build and programmed by most non-expert users.

Fig. D.4(a) shows the modules as an exploded view of the Quadruped shown in Fig D.4(b). The quadruped consist of eight passive modules: four feet, two 'Y',



(a) Blockly

```
api.startup()
position = None
api.speak(str('My name is Fable'))
while True:
  position = api.getModuleMotorPosition(11, 1)
  api.setModuleMotorPosition(13, 0, position)
  api.plotDataAppend(position, label ='A', windowSize=50)
api.sleep(1)
api.terminate()
```

(b) Python

Figure D.5: (a) A simple example Blockly program for controlling two Fable joint modules. (b) The corresponding Python code output by Blockly.

one 'X' and one tail module. In addition, four active joint 2DOF modules is used in the quadruped. It takes on average 16.9 seconds to assemble and 9.8 seconds to disassemble the robot for an experienced user. This type of quadruped robot has previously been programmed by young students with different gait patterns, who in the process learned about programming, robotics, and mathematics.

A simple example of a Blockly program, which can be used to control the robot, is shown in Fig. D.5(a). This program first uses a web API to speak a sentence (local playback on the PC) and then enters an infinite loop where it mirrors the angle of one joint on module 11 to the opposite joint on module 13. Further, the program plots the position in real-time in a graph which is an integrated part of the graphical user interface for debugging and educational purposes. This program enables the user to use one joint actuator to remote control another joint on a different module. We have observed such simple interactions to be both playful and engaging for younger children (age around 9-10 years), who are also able to program such behaviors themselves. Tests have shown that the main loop of this program is executed with 90 Hz on average. The main limitation on the execution speed is the communication path from PC to dongle (serial), dongle to module (radio), module to Dynamixel servo (half-duplex serial) and back again in the case of the 'get' method.

In Fig. D.5(b) the equivalent Python source code generated from the Blockly program is shown. We anticipate that the similarity between Blockly and Python will facilitate non-programmers to transition from a visual programming language

to more conventional programming. More advanced control strategies e.g. for research experiments could be implemented in Python, Java and in future work also in Matlab.

D.6 Conclusion

In this paper we briefly described the vision, concept and design of the latest version of Fable. Fable is a modular robotic playware platform that enables non-technical users, ranging from young students, makers and researchers to assemble and program their own interactive robots. We described the mechanics of the system with passive and active modules and a scalable and robust connector design, that enables users to build and reconfigure a robot in a matter of seconds. The paper also provides an overview of our modular electronic design and how this allows us to build and develop new types of modules faster. Further, we described how the user could program the system as if it was a centralized robot at different levels of abstraction ranging from a visual programming language to conventional languages such as Python and Java. Currently, we are extending the Fable system with more passive and active modules, including a gripper, a head module with various sensors and a rotational base module which can also be used as a wheel. Furthermore we are working on interfacing Fable with Matlab and Simulink to allow researchers to simulate algorithms with hardware in the loop.

Paper E

Fable II: Design of a Modular Robot for Creative Learning

E.1 Abstract

Robotic systems have a high potential for creative learning if they are flexible, accessible and engaging for the user in the experimental process of building and programming robots. In this paper we describe the Fable modular robotic system for creative learning which we develop to enable and motivate anyone to build and program their own robots. The Fable system consists of self-contained modules equipped with sensors and actuators, which users can use to easily assemble a wide range of robots in a matter of seconds. The robots are user-programmable on several levels of abstraction ranging from a simple visual programming language to powerful conventional ones. This paper provides an overview of the design of Fable for different user groups and an evaluation of critical issues when we attempt to integrate the system into an everyday teaching context.

E.2 INTRODUCTION

Today's world is filled with consumer products that constantly encourage us to buy and not to build. Taught to us from an early age, plagiarism and copyright policies serve as mental barricades that dry out our curiosity, creativity and collaboration [1]. In this work we seek to revitalize and quench our users thirst for knowledge within the domain of robotics. We believe that, given the right tools, anyone can become a robot designer.

In this paper[1] we present the design of Fable, a mechatronic construction kit that allows users to playfully build and program their own robots. Our objective is to motivate users both towards making their own robots and sharing them with others, that is as a DIY (Do it Yourself) kit and as a DIT (Do it Together) kit.

We have designed Fable as a modular robotic platform with a focus on the users' needs, ranging from a classroom of kids, and after-school clubs, to hobbyists/makers and even researchers, as illustrated in Fig E.1. This diversity in usability is achieved by encapsulating key robotic functionalities into modules that can be combined in numerous configurations utilizing a shared connector and communication system. This gives us the freedom to design basic modules for kids and high-end modules for researchers while making it easy for makers to start building their own. To support this diversity of users we enable them to program the system by using their preferred programming language (Blockly, Python and Java are currently supported and we plan support for Matlab).

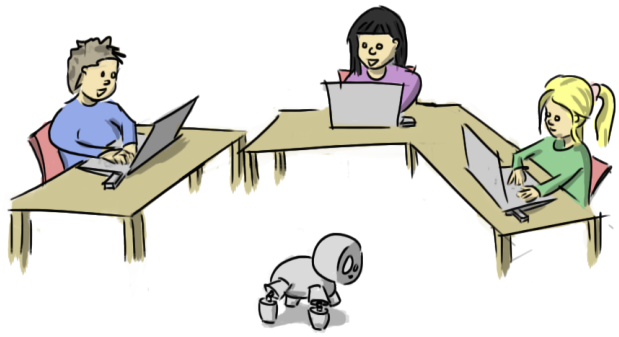
The rest of this paper starts by describing related work (Sec. E.3). It continues by presenting the design of Fable, that is: mechanics, electronics and software (Sec. E.4). Further in Sec. E.5 the paper exemplifies how robots can be assembled in seconds, programmed with Blockly and Python, and we evaluate Fable as an educational platform based on programming sessions with users and experts.

E.3 Related Work

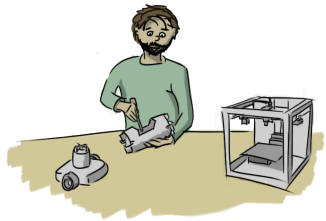
The Fable system is designed to support the user's creative thinking and innovation. In order to guide the development of creative toolkits for users, Resnick et al. has proposed a set of design principles such as "Low Threshold, High Ceiling, and Wide Walls" [135]. Von Hippel described toolkits for user innovation as a way to transfer design abilities from the toolkit developer to the user [138]. Further, Von Hippel proposed five key objectives for such toolkits: 1) Enable the user to perform trial-and-error learning, 2) span a solution-space, that embraces what the user wants to design, 3) is user-friendly by being familiar and easily accessible to the user, 4) contains a standard library, that users can combine with their own designs, 5) automatically translate the user's design into the format required to produce the design. In this work, we are guided by such design principles and objectives in order to make Fable as valuable as possible for its users.

Fable is a modular robotic system. Such systems achieve flexibility and versatility through modularity and thereby provides users with what is known as "Wide Walls".

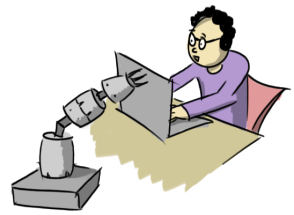
[1] An earlier version of this paper was presented at the IROS 2014 workshop on Modular and Swarm Systems [6]



(a) Students



(b) Maker



(c) Researcher

Figure E.1: Prospective Fable users with different interests, objectives and level of experience.

Modular robotic systems consist of a collection of simple robotic units that can attach and detach from each other to form a wide range of configurations [15, 62]. While the majority of modular robots are designed to study self-reconfiguration, e.g. [94, 19, 30], Fable takes its inspiration mainly from user reconfigured and interactive systems. Fable is more similar to Roblocks/cubelets [60], MOSS [134], Topobo [57] and LEGO Mindstorms in that all these systems aims to enable and motivate everyone to become a robot designer.

With Fable we aim to provide a creative experience to its users. Dahl and Moreau defined experiential creation as "*...activities in which a consumer actively produces an outcome*" [139]. For robots this definition ranges from kits which the user assembles into a specific predefined robot to open-ended systems where little or no guidance is given to the user in how the robot should be designed. The modularity and granularity of Fable provides constraints, and thereby guidance, to the solution-space but the system is prepared for makers to overcome such constrains, e.g. by creating their own types of modules.

Ideally, Fable should motivate the users to be creative and learn in the process.

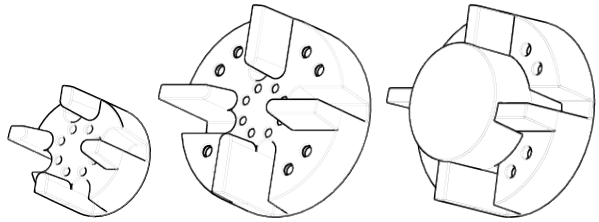
As observed by Dahl and Moreau [139] users are motivated to engage in creative work for both intrinsic and extrinsic reasons, including a feeling of accomplishment, a desire to learn and to share a creative experience with others. Further, users are motivated by the satisfaction they feel from an immersion in the creative process [139]. This immersion is related to the mental state of flow which is characterized by a lost sense of time and being fully absorbed in the current activity [136]. A state of flow is more likely when the user feels that the activity i) has a clear goal, ii) has immediate feedback on the users performance and iii) has an appropriate balance between the challenges of the task and the user's perception of own skills [136]. Play share many characteristics with flow and, as argued by Brown, is ubiquitous in nature and critical for both children and adults development and well being [140]. Play can motivate users to perform learning activities and is therefore a key objective when designing interactive learning environments [141]. The PlayGrid is a model which aims to encapsulate what makes a user enter a state of Play, based on four types of play: the Assembler, the Director, the Explorer, and the Improviser [142].

Educational robotic kits such as Lego Mindstorms and VEX are widely used in classrooms for their mechanical flexibility and their easy to use programming environment. These kits, including Fable are inspired by the learning theory of constructionism [143] and aim to move users away from the passive individual thinking and into active hands-on collaborative learning-by-building. An important difference between Fable and fine granularity systems, such as Lego Mindstorms, is the reduced effort required by the user to modify and experiment with their mechanical design.

A visual programming language enables a "Low Threashold" entry to robot programming. For Fable we utilize Blockly [90], a block programming approach similar to Scratch [137], to allow non expert users to graphically program and interact with the system. In order to address the needs of different users, to provide scaffolding and a "High Ceiling" also Python, Java and eventually Matlab can be used to program the system.

E.4 Fable Design

This section provides an overview of the design of our second version of the Fable system, details about the first version can be found in our previous work [3, 4, 5].



(a) Small size (b) Medium size (c) Compatibility of two connector sizes

Figure E.2: Connector design: With the current design any size is possible and compatible with the rest, having only as a lower limit the small size connector diameter.

Mechanics

Our approach uses powerful, yet easy to connect modules that allow users to assemble a functional robot in a matter of seconds. The Fable system is divided in active and passive modules. Active modules contain a set of electronic boards with a microcontroller, onboard power, and a radio device for wireless communication with a PC. These modules also provide functionalities through actuation and sensing, e.g. one active module design is a 2 degree of freedom joint, see Fig. E.4(a). Passive modules consist of a variety of shapes made out of empty plastic shells. These passive modules help give the robot structure and shape, e.g. a 'Y' shaped module is used to connect three modules together and an 'X' to connect four. Both can be seen in the robot configuration in Fig. E.4(b).

A key feature in modular systems are the connectors since they serve as the only contact surface between modules. Our current connector design, illustrated in Fig. E.2, is genderless and four way redundant, which allows our users to explore several connection possibilities between modules. Each connector has at least one ring of magnets that attaches to a matching set on the connecting end. The connector uses a set of flanges that lock the modules allowing only the user to disconnect them by pulling them apart. With this design we obtain a strong connection between modules and yet it's easy enough for children to disconnect. The connector design is scalable, meaning that it is compatible between different sizes, giving us the possibility of combining large modules with small ones, as illustrated on Fig. E.2(c).

Electronics

For the Fable system we have developed a set of electronic boards, that when combined with commercially available boards give us a modular electronic con-

figuration. The electronic boards are designed for simplicity, low production cost, flexibility and hackability. The modular approach enables us to create different active modules by mixing electronics boards in new configurations. Table E.1 describes the electronic boards currently used in Fable. Different active modules and a radio dongle will use a specific subset of the electronics modules. To facilitate hackability the module and dongle firmware is executed on an embedded Arduino board. As we develop new types of Fable modules we will develop new modular electronic boards to support them.

Board Name	Details	Description
Module Board	11.1v, 1000mAh LiPo battery	Main mother board for active modules
Dongle Board	5v USB powered	Main mother board for dongles
User Interface Board	RGB diode, button, buzzer, recharge plug, on/off switch	Functions for user feedback and direct interaction with module/dongle
Motor Board	RS232 half-duplex dual buffer	Interface board for serial control of Dynamixel motors (e.g. AX-12A)
Arduino Pro Mini	AVR Atmega328, 3.3v, 8MHz	MCU module for dongles and modules
Radio Board	NRF24L01+, 2.4GHz, 2MBit, SPI interface	Wireless communication between modules and dongles

Table E.1: Overview of electronics boards used in Fable

System Network Architecture

The underlying objective of the Fable network architecture is to make the robot programming as simple and flexible as possible. Due to a low lag radio communication link to the modules, the user can program the distributed robots as if it was centralized and connected directly to the PC. Therefore, the user avoids the difficulties of cross-compiling, downloading program to robot and debugging a distributed embedded platform.

The network architecture of the Fable system is shown in Fig. E.3. The user PC is serially connected to a dongle which provides a shared 2 Mbit radio communication link between the user controlled application and the modules. Modules are addressed using an ID and their module type. Web services can be used to enhance the possibilities of the user application, currently we use a web service API for speech generation. Although the radio communication is shown here as a master-slave architecture, it can also function as a peer-to-peer network. This can be exploited in certain research studies, e.g. on distributed control as described in

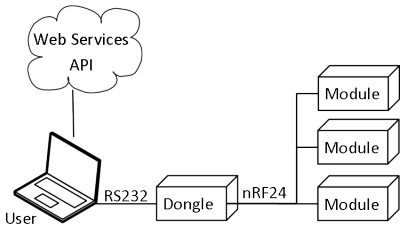


Figure E.3: Network architecture of the Fable system.

Section E.4. In addition, we plan to exploit asynchronous communication between different dongles to enable the different users to program collaboratively by writing different parts of a program and remotely calling functions developed by other users.

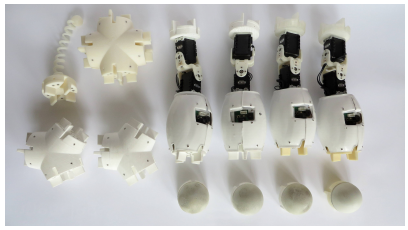
Distributed Hardware-in-the-loop Control

Research within modular and swarm robotics often explores distributed control strategies, e.g. for behaviors such as locomotion, learning and shape-formation [144, 145, 46]. Debugging and validating distributed controllers is challenging and hardware-in-the-loop simulators can facilitate the development process [146].

Fable is mainly centralized controlled given that it receives commands wirelessly from a PC and lacks dedicated neighbor-to-neighbor communication. However, Fable can also be used as a hardware-in-the-loop simulator for developing and validating distributed control strategies. The software API supports distributed control where each module controller executes in a separate thread on the PC. Also, passive modules will run independently in a control thread to take part in the communication network. Given an adjacency matrix of the robot the modules can simulate neighbor-to-neighbor communication. Fable supports two modes of distributed communication:

Simulated Messages are exchanged between module control threads locally on the PC. Transmissions are performed without significant lag, without payload length limits and messages are never lost.

Hardware-In-The-Loop Messages are wirelessly transmitted between two real-world modules for a more realistic performance. The lag of transferring such a message varies from approx. 6 ms to 13 ms with the length of the payload (0 to 26 bytes respectively). Message loss heavily depends on the specific setup, for



(a) Modules

(b) Quadruped

Figure E.4: (a) Four active joint modules and eight passive modules. (b) A Fable quadruped robot assembled from the above twelve modules.

example in one setup with 15 meters between two modules we measured package loss to increase from $4.6 \pm 2.7\%$ to $43.4 \pm 22.6\%$ when a human moved within the line-of-sight between the modules.

User Programming

The user develops the system application from a personal computer using one of the supported programming languages. When the user executes the application program it is not cross-compiled but run locally on the PC. Simple APIs enable the user-program to call functions on the remote modules through the dongle connected to the PC (based on module IDs). The round-trip lag for a simple remote procedure call is around 4.2 ms, longer if the remote module needs to perform processing. This is sufficiently low for most applications and can be reduced even further in future work.

We developed a Graphical User Interface (GUI), based on Blockly [90], as a means for non-programmers to quickly start developing applications. Blockly is an open-source framework for building application specific visual programming languages inspired by the Scratch programming environment [137]. An example

of a Blockly program for controlling a simple interactive Fable robot is shown in Fig. E.5(a). The source output generated from Blockly is written in Python. This string of Python code is send from the Blockly JavaScript web-application through an HTTP request to a Python server which executes the Python control application. Further, the Blockly interface enables users to collect and analyze data from the robot, e.g. as real-time-plots of sensor values.

More experienced users can also write applications directly in either Python or Java based on APIs. Further, we plan to add Matlab support for research and university education in future work.

E.5 Examples and Evaluation

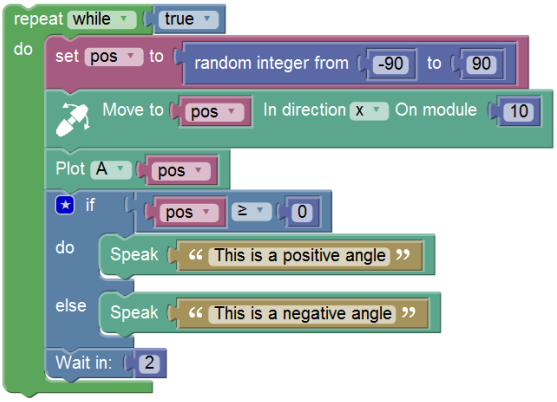
This section describes an experiment to measure the assembly time of a robot, present an example of Blockly programming and present qualitative observations based on programming sessions with Fable.

Assembly and Disassembly Speed

This example illustrates the assembly of a typical robot that could be built and programmed by most non-expert users. In programming sessions students have previously programmed similar quadruped with different gait patterns, and in the process learned about programming, robotics, and mathematics. In such sessions, we have observed the importance of fast and easy reconfiguration which motivates the students to explore and experiment with the robot morphology.

Fig. E.4(a) shows the modules as an exploded view of the Quadruped shown in Fig E.4(b). The quadruped consist of eight passive modules: four feet, two 'Y', one 'X' and one tail module. In addition, four active joint 2DOF modules are used in the quadruped.

We have measured that it takes on average 16.9 seconds to assemble and 9.8 seconds to disassemble the robot for an experienced user. This corresponds to 1.5 second for connecting two modules and 0.8 seconds to disconnect. For comparison, Davey et al. reported on the assembly time of CKBot with the ModLock connector which took 7 seconds per connection and 3 seconds per disconnect in a snake configuration [147]. Compared to many other systems, both Fable and CKBot are fast to assemble, e.g. it takes 1917 seconds to assemble a simple Bioloid snake, as reported by Davey et al. [147].



(a) Blockly

```
api.startup()
import random
pos = None
while True:
    pos = random.randint(-90, 90)
    api.setModuleMotorPosition(10, 0, pos)
    api.plotDataAppend(pos, label ='A', windowSize = 50 )
    if pos >= 0:
        api.speak(str('This is a positive angle'))
    else:
        api.speak(str('This is a negative angle'))
    api.sleep(2)
api.sleep(1)
api.terminate()
```

(b) Python

Figure E.5: (a) A simple example Blockly program for controlling a single Fable joint module (ID=10). (b) The corresponding Python code output by Blockly.

Visual User-Programming

A simple example of a Blockly program, which can be used to control Fable, is shown in Fig. E.5(a). This program controls a joint module to random angles and reports to the users which angles are positive and negative while illustrating the angles on a plot. Students can build such a program as an exercise to help them visualize angles in different contexts: plots, motors, degrees.

The program runs in an infinite loop where it starts by assigning a random integer to a variable. The variable is then used as an angle set-point on the joint module with ID number 10. Then it continues to plot the motor's angular position and evaluates if the position is positive or negative and speaks its correspondent sentence. The speak function uses a web API to speak a sentence, with local playback

on the PC. Further, the plot function plots the position in real-time on a graph which is an integrated part of the graphical user interface. Plotting is useful for debugging and educational purposes.

In Fig. E.5(b) the equivalent Python source code generated from the Blockly program is shown. We anticipate that the similarity between Blockly and Python will ease beginner programmers' to transition from a visual programming language to a general purpose programming language. More advanced control strategies e.g. for research experiments could be implemented in Python, Java and in the near future also in Matlab.

Programming Sessions

In order to evaluate Fable as an educational platform we arranged a number of programming and building sessions with educational experts, schoolteachers, and students from different age groups (ages 8 and up). Some of the sessions were combined with semi-structured interviews. The focus was on late primary and secondary school in order to learn how the system could be integrated in a school context. For practical reasons we focused our attention on the Danish school system but we anticipate that our observations may generalize to other school systems as well. Based on these sessions we identified critical themes that will guide the further development of Fable:

Classroom integration The structure of everyday teaching in schools affects how robotics can be integrated in the classroom. In the Danish school system the classes are organized in 45 minute lessons. In a typical lesson the teacher first lectures the topic followed by students working on exercises related to the topic. In order for robotics to fit in this time-scarce context, it is important that the setup time of the system is kept to a minimum. However, most robotic kits require more time in the assembly phase, e.g. the LEGO Mindstorms Education EV3. A minimal EV3 mobile robot requires the user to follow a 40-step assembly manual (equvivalent to estimated 20-40 min.) [148]. Mechanical assembly arguably has valuable learning outcomes, but in practice it limits the use of robots to longer integrated projects or the use of pre-assembled fixed morphology robots. In order to address this issue, in the design of Fable our objective is to keep the setup and assembly time as low as possible.

Educational material alignment Creating and programming robots freely based on intrinsic motivation can be a highly rewarding and educational experience for

the students. However, in order to achieve acceptance within the conventional school system educational material must be designed in order to realize specific learning outcomes. In the Danish school system such learning objectives are defined at a national level, but how to teach them is left up to the individual schools. Most schools base their teaching on a standard text books which are written for a specific grade covering one class. For Fable the most relevant classes are Math and Physics, and to a lesser extent Biology and English. Therefore, in order for Fable to fit in a classroom context educational material must be developed that is closely aligned with the national learning objectives. We anticipate that such educational material will most likely lead to the development of new Fable module types.

Motivating teachers Robotic toolkits have the potential to provide engaging hands-on learning experience to students. However, as noted by Bers et al., few teachers have the experience and skills necessary to integrate such toolkits in the classroom [149]. This stresses the importance of designing an easily accessible robot system, training the teacher in their usage and providing them with familiar materials that will make the learning process for them both enjoyable and rewarding. It is a primary concern of the Fable system to motivate the teachers and give them the necessary confidence to use the system in the classroom.

Motivating students Students enjoyment is affected by how the learning situation is structured. In one extreme the students follow a strict tutorial to build and program a specific robot. At the other extreme the students are left without any constraints to build whatever they want. Dahl and Moreau found that providing users with instructions and a general goal, but no specific target, would result on a higher feeling of competence, autonomy and task enjoyment compared to situations with no instruction or a specific target [139]. In the context of Fable, we have observed that a lack of instruction and clear goals are likely to quickly make the students frustrated. On the other hand, we have also observed how good instruction combined with clear goals such as challenges, competitions and performances made the students highly engaged in the activity. How to integrate such motivating activities in a time-scarce context with well defined learning objectives is a critical topic for further research.

E.6 Conclusion

In this paper we described the design of the second version of Fable. Fable is a modular robotic platform that enables non-technical users, ranging from young

students, makers and researchers to assemble and program their own interactive robots. We described the mechanics of the system with passive and active modules and a scalable and robust connector design, that enables users to build and reconfigure a robot in a matter of seconds. The paper also provides an overview of our modular electronic design and how this allows us to build and develop new types of modules faster. Further, we described how the user could program the system as if it was a centralized robot at different levels of abstraction ranging from a visual programming language to conventional languages such as Python and Java. We also described how the system can be used as a hardware-in-the-loop simulator for research in distributed control strategies. We evaluated the robot and found that a typical configuration could be assembled or disassembled in less than 20 seconds which is important to motivate users for trial-and-error learning and for classroom integration. Further, we identified critical themes for integrating Fable in a school context based on programming sessions with different user groups. In future work we will continue to optimize the Fable system to meet the requirements of its users. Currently, we are extending the system with more passive and active modules, including a gripper, a head module with various sensors and a rotational base module which can also be used as a wheel. Furthermore we are working on interfacing Fable with Matlab and Simulink to allow researchers to simulate algorithms with hardware in the loop.

E.7 Acknowledgments

This work was performed as part of the “Modular Playware Technology” project funded by the Danish Advanced Technology Foundation as well as additional funding from the Dr. Techn. A.N. Neergaard og Hustrus foundation and the Siemens Foundation. Thanks to all the members of Center for Playware for their contributions to the project.

Paper F

Fable I: A Modular Robotic Playware Platform

Abstract

This paper provides an overview of Fable which is a user-reconfigurable system composed of heterogeneous robotic modules. The system is designed for users to easily create their own robots. The Fable system consist of partly self-contained joint modules, branching modules and termination modules. Joint modules are equipped with two servo motors for developing mobile robots. A unique connector design allows users to build a robot in a matter of seconds. We present several example applications with users interacting and programming the Fable system.

F.1 Introduction

Today's manufacturing world is changing, production is moving back from the industrial manufacturing parks and into private communities. This is achieved thanks to the success of rapid prototyping technologies as well as low-cost and flexible industrial robots.

In this changing world of we consistently ask ourselves: Can we build a system capable of nourishing people's curiosity and enable users to build a strong entrepeneur self-esteem? To explore this question we have designed the Fable system as a modular robotic kit that allows users to easily assemble and modify their robotic constructions.

We believe that people are able to learn more and achieve high levels of concentration while capturing a user in a deep mental state of play. A good example of

this are construction toys (e.g. LEGO). In this work we explore modular playware as mechatronic construction toys that are animated and which become playfully alive as they are assembled from individual modules. In order to achieve this, we are developing novel interactive technologies by mixing experiences from modular robotics, embodied artificial intelligence, and human-robot interactions.

Modular robots consist of a set of simple robotic units that can attach and detach from each other to form virtually endless different configurations [42, 62]. Even though, many systems are designed to support self-reconfiguration, e.g. [94, 19], other systems are designed for rapid robot prototyping [95], constructing walking robots [96], adaptive furniture [97], or space exploration [43].

In this paper we present the mechanical design of the Fable modular robotic system as well as some applications of the system. Thanks to a unique design users can combine Fable modules in many ways to create various robotic creatures, such as snakes, walking robots, vehicles, humanoids, or even fantasy creatures. The long-term vision is to transform the development of robots from something done solely by experts to something so widely available, easily accessible, and motivating, that anybody is able to realize their ideas to life or bring solutions to problems encountered in their own lives.

We also explore modular robots as a means of robotic playware, that is modular systems designed to enable user to build their own robots just for the fun of it. During the development of Fable modules we took inspiration from some available robotic systems such as Topobo a system that enables users to record and playback motor sequences by exploiting the programming-by-demonstration approach [57] and the roBlocks (now Cubelets) which use a programming-by-building strategy where the behavior is emerges from the interaction modules with each other and the environment [61]. Further, the LEGO Mindstorms is a robotic construction kit that enables direct-user-programming of LEGO models equipped with actuation and sensing. Similar to these systems Fable aims to enable everyone to become a robot designer and become motivated to be creative, explore, construct, reflect, iterate, play and share.

This paper explores modular robotic playware, which are modular robotic systems designed to enable a user to construct artifacts for playful activities [98]. The modular robotic playware approach suggests that, the best way to allow users to develop robotic systems is through contextualized hands-on problem solving, which permits the users to work directly with technological building blocks in their own context. By the free manipulation of combining the technological building blocks, the user is developing technological prototypes him/herself, and these technological prototypes can be tested immediately as they are being constructed by the user

in the user's context [99]. In general, the modular system becomes an object to think with, and the modularity invites the user to perform physical manipulation and reconfiguration [100]. To facilitate human-robot interaction we are inspired by robots such as Probo [101], Huggable [102], and Kismet [103].

We expect that the Fable system, when fully developed, will complement the previous playware examples in three novel ways: i) The possibility to build new robots with powerful actuators and sensors in a matter of seconds, ii) Robots can be programmed in several ways depending on the user's preference (e.g. programming-by-building, programming-by-demonstration, programming-with-tags, and direct-user-programming), iii) To easily develop socially interactive and adaptive robots by using smart sensor modules that provide higher-level information about the user.

In the rest of this paper we describe the mechanical design (Sec. F.2) and the electronic design (Sec. F.2) of the Fable system. We present the Software Architecture in Sec. F.2 and then follow with two example applications in Sec. F.3. We continue in Sec. F.3 by describing a series of user tests and we conclude this paper in Sec. F.5.

F.2 Design of Fable

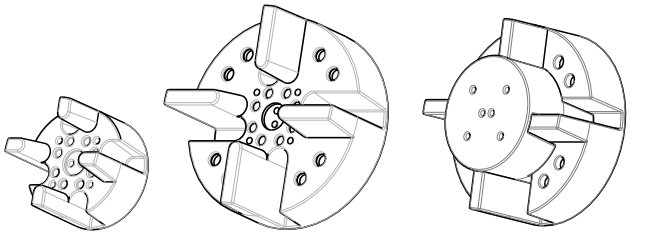
This section gives a general description of the the topology of Fable and it explains the decisions behind the key design aspects of the system. Some of these aspects include connector design, module types, module design, user programming and software architecture.

Heterogeneous, Scalable, Chain-based Topology

The Fable system is designed as a modular robotic playware platform, suitable for creating interactive creatures that can be used as interactive toys, hobby projects and even research platforms.

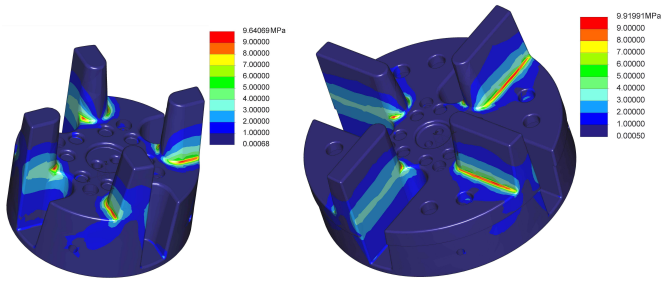
We decided to make Fable a chain-based system and not a lattice based system, in order to simplify the assembly of functional robots [62]. A Fable robot consists of a set of heterogeneous modules that provide the necessary functionality to be able to perform various tasks such as sensing, actuating joints for movement and producing sounds. Throughout the design process our objective was to keep it simple, reliable and appropriate for interaction with non-technical users including children.

Mechanical magnetic connectors allow a solid attachment and detachment between modules for rapid construction of robots. The morphologies can be rear-



(a) Small connector (b) Medium connector (c) Compatibility of two different sized connectors

Figure F.1: Connector design: With the current design any size is possible and compatible with the rest, having only as a lower limit the small size connector diameter.



(a) Small connector (b) Medium connector

Figure F.2: Von Misses Stress analysis on a PLA design with an applied pressure of 0.45 MPa per flange on the medium size connector and 0.55 MPa per flange on the small size connector.

ranged in numerous configurations based on joints, branching, and termination modules. Thanks to the rounded and organic shapes, Fable modules are aesthetically pleasing. Further, the connectors are scalable, this means that users can combine modules of different sizes.

Connectors

In modular systems, connectors are the only physical interface between neighboring robots. They are responsible for maintaining a given configuration, as well as for allowing it to change. The following list sums up the requirements for the connector design.

- Strong connection and yet easy enough for children to disconnect.
- Robust to wear and tear.

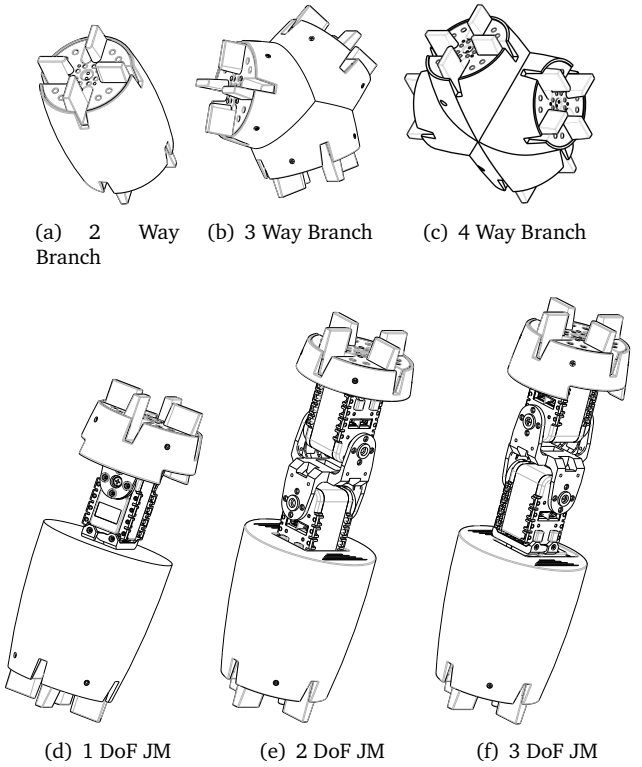
- Transfer communication signals.
- Genderless.
- Easy to Manufacture.
- Multiple connection configurations.
- Scalable to maintain compatibility with larger and smaller modules.

Figures F.1(a) and F.1(b) show the overall design of two connector sizes: small and medium. The connectors have a repetitive pattern every 90°, this pattern allows us to attach connectors in four possible ways. Each connector has at its center two holes that enable IR communication between modules. The IR holes are placed in an indentation of around 2 mm to allow the IR beams to spread and thus establish a communication regardless of the orientation used to attach both connectors. The IR communication was not used in the presented prototypes.

Furthermore each connector has at least a ring of magnets 8, where each 90° wedge of a connector contains two magnets with alternating S-N poles, a whole and a flange. These patterns allow users to attach connectors in 4 possible orientations. In this iteration we produced two connector sizes: medium and small. The medium size contains two sets of concentric magnetic rings, a large one to ensure compatibility between same size connectors and a smaller one to extend its compatibility to small sized connectors, see Fig F.1(c).

We've implemented flanges to mechanically lock connections against twisting and bending, thus giving users only the possibility to detach modules simply by pulling them apart. In this way the magnetic force is concentrated on the same axis as the disconnection axis, which enables a strong connection between modules, requiring a force of 12 N or 30 N to disconnect the small or medium size respectively. Through this design we allow users as young as 6 years old to easily build their own Fable creations.

The European Standard for Safety of Toys requires a toy to withstand a torque of 0.34 Nm for a ten second interval in a clockwise manner as well as in the opposite direction [106]. We made a stress analysis where a pressure of 0.55 MPa is applyed per flange on a 56 mm^2 area and 0.45 MPa per flange on a 196 mm^2 area on the small and medium size connectors respectively. The results on Figure F.2 show that it is necessary to apply a torque of 2.094 Nm or 8.11 Nm in order to break the mechanical lock of the small or medium size connector respectively. Furthermore, as a tensile test the European Standard requires a dead weight of at least 90 N



(a) 2 Way Branch (b) 3 Way Branch (c) 4 Way Branch

(d) 1 DoF JM (e) 2 DoF JM (f) 3 DoF JM

Figure F.3: Fable Modules

when the largest accessible dimension is greater than 6 mm. Both requirements are fulfilled by the design.

Fable Modules in Version I

Fable consists of two types of modules: branching and termination. For future iterations we plan to extend the system with more actuated modules such as a wheel module as well as with various types of sensor modules.

subsubsectionBranching modules (BM) are units used to connect several modules (2 or more), together in tree-like configurations. We currently have four designs of Branching Modules. The first one being the 2 Way Branch, that can be either active or passive (see Figs F.3(d)-F.3(f) and F.3(a) respectively). These modules enable communication between both ends. The second type is the 3 Way Branch, shown on Figure F.3(b), which establishes connections and communication at 120° intervals. The 4 Way Branch, as seen on Figure F.3(c), connects modules with an

offset of 90°.

Joint Modules (TM)

are a type of active branching module. They include a series of daisy chained AX-12A Dynamixel motors. For this iteration we present three designs (Figs F.3(d)-F.3(f)), although only the 2 DoF version was physically implemented.

Termination Modules (TM)

are designed to close off open connectors on a robot and provide the system with functionality. Termination Modules, just as Branching Modules, can be either active or passive. These modules have the potential to add visual expression, additional sensors, or actuators (e.g. grippers or wheels). We are currently developing several termination modules, including a foot module, to enable walking creatures to walk more efficiently; a wheel module and a vision module, which makes use of Asus' Xtion PRO LIVE to equip the system with stereo vision.

On Table F.1 we summarize the module characteristics of the current version of Fable and we specify which modules were implemented and tested. F.3.

Module	Height (mm)	Weight (g)	DoF	Implemented
2 Way Branch	142	300	0	Yes
3 Way Branch	130	350	0	Yes
4 Way Branch	142	450	0	Yes
1 DoF JM	210	450	1	No
2 DoF JM	244	500	2	Yes
3 DoF JM	275	550	3	No

Table F.1: Module Characteristics, all weights given are including battery and electronics

Electronics

Main board

Each active module is meant to have its own electronics. The main board has an Atmel ATMega2561 microcontroller running at 8 MHz with 256KB of FLASH and 8KB of RAM. The boards have a connector to power and can control several daisy chained AX-12A Dynamixel servos. Further, the boards support up to four IR channels (half-duplex) used for neighbour-to-neighbour communication. In addition, each board can be equipped with an XBee dongle for wireless communication between modules or a PC. All of the electronic boards have an accelerometer, a gyroscope, a buzzer and the possibility to attach additional sensors. The boards are powered from a three cell 11.1V lithium-polymer rechargeable battery with a

capacity of 1000 mAh. The microcontroller can be programmed externally without disassembling the modules. These custom made electronic boards are in the process of being integrated into the system.

On the presented version of the system, we used a compatible embedded system, CM-510 controller box, as a means of building a modular system with centralized control of the modules. However, the software and control architecture used is fully functional and can be ported directly to the electronic boards for distributed control.

USB Host

During the development of the system we introduced a CM510 controller box into a 4way Branch and used male-male USB cables to transmit signal and power to other modules, allowing us to have as a middle step a system with modular mechanics but centralized control.

We added a USB host to allow easy reprogramming for non expert users. This interface allows non expert users to feel comfortable and use a familiar way of inputting data to a device. The USB stick must contain only one application in the root of the drive, since Fable will load, upon start-up, the first application it finds.

The USB Host consists of an Arduino Pro Mini, running at 5V and loaded with an ATMEGA328P and the VDIP1 prototyping board. A virtual Machine developed by Brian Silverman [150], gives Fable the possibility to load Pico Logo applications upon start up when stored on a USB stick.

User Programming and Software Architecture

Developing applications for Fable should be open and accessible to non-expert users (e.g. designers, students and even kids). Hence it is of key importance that our programming tools hide away many of the low level details of sensors, motors and distributed processing. Therefore we have decided to split the Fable's software architecture in two levels: a low-level firmware to handle details of sensors, motors and distributed processing, and a virtual machine to provide high-level programming of user defined applications.

The microcontrollers in Fable are run a low-level software system primarily developed in C. In order to ease the development of the embedded firmware we use the Assemble-and-Animate framework (ASE) [107]. This gives us a high level of abstraction, a large library of components and algorithms and an asynchronous event based framework.

C programming can be too complex for inexperienced programmers, therefore we chose LOGO as an alternative language for application development [52].

Fable's firmware includes a virtual machine (VM) that runs as a (non-blocking) process in the ASE scheduler and can execute LOGO applications compiled to byte-code. The VM not only makes it easier to program Fable but its also safer to run custom applications, since it limits the program's access to the systems resources.

We use a restricted version of LOGO for embedded devices, called PicoLOGO developed by Brian Silverman [108]. PicoLOGO has a limited set of instructions and is restricted to integer type variables. Although the biggest downside when using PicoLOGO is speed, as microbenchmarks indicate that it is 9 times slower than C.

For more advanced processing, than possible or feasible on an 8-bit mirocontroller, we use the Robot Operating System (ROS) [109]. ROS gives us access to a large set of software libraries such as OpenCV. The software architecture is such that we run ROS on a server which provides services (over a wired or wireless serial connection) to the LOGO code running on the modules. This architecture has for instance enabled us to develop a humanoid model, from the modules, that can speak, detect faces, and mimic postures from humans using a motion sensing input device [4].

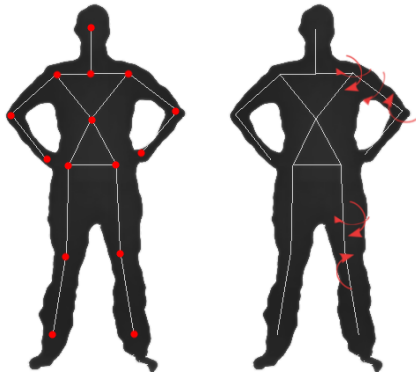
F.3 Example Applications

High-School Workshops

Interaction with 3D Vision Sensor

An efficient way for humans to communicate is verbally, but a lot of what we say can be inferred from our body language (e.g. body posture, gestures, facial expression and eye movements), and even contradict our verbal expressions [151]. Body language can provide clues to attitude or state of mind of a person, or simply to give commands or descriptions. We decided to experiment with the social interactivity of Fable and enable the system to detect human postures. We consider a posture to be a set of labeled angles, where each angle represents the orientation of a user's joint. In this implementation Fable is able to detect postures from a predefined set stored in a server's database. In this implementation we used a 3D vision system for posture detection and face recognition, we also built a humanoid torso, composed from a 4 way branch as the torso, a 2 DOF JM for each arm and an Asus Xtion Live Pro camera, similar to the more popular Kinect, as a head.

The camera provided a set of user postures for all the major limbs in the body.



(a) Red dots showing joint positions available for each user.

(b) Angles of arms and legs that could be calculated from the joint positions.

Figure F.4: Image showing joint positions and angles available for each user.

Fig. F.4 shows the limbs and joints detected by the vision system. As a basis for tracking and detecting users we use OpenNI. Furthermore we also implemented face recognition as a first step to have Fable adapt to different users personalities and ages. To achieve this first we start by taking an image of a user and extracting his face. We then continue to extract keypoints from the extracted image using the Features from Accelerated Segment Test algorithm (FAST) [121, 152].

We then give a description using the Scale-invariant feature transform algorithm (SIFT) and then follow to compare the descriptors to the database using the Fast Approximate Nearest Neighbor matcher (FLANN) and if there's a match an event is triggered. In our implementation the server printed out the name of the user. For further details on the implementation of the vision sensor refer to our previous publication [4].

Playful Voice Control

In a search towards interacting with robots in novel ways we introduced a voice control, that unlike the ones commercially available, is language independent. That is kids from different age groups and cultures are able to interact with Fable using the same application. The sensor uses the different pitch levels to determine wether the robot should move forwards, backwards, turn left or right.

Since our goal is to develop inexpensive modular sensors, the algorithm used for the voice sensor can be implemented in any 8bit microcontroller. Furthermore

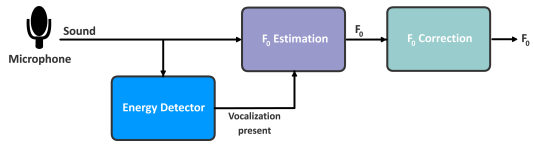


Figure F.5: Elements in the F_0 sensor.

in the current implementation the sensors processing is being done by a computer, which adds some latency giving a near-real-time response.

The voice control sensor consists of three major elements (see Fig. F.5). The first one being an energy detector, used to determine if the user is vocalizing. If speech is detected, the second element the signal's F_0. In speech F_0 refers to the fundamental frequency or the frequency at which the vocal chords open and close. The F_0 correction, monitors the F_0 estimates and removes outliers caused by noise or unvoiced sounds.

When estimating F_0, we currently assume that the signal is always voiced, even though not all speech is voiced. In the future, we plan to add a harmonicity detector to determine if the vocalizations produced are voiced or not.

Energy Detector

A simple approach to detecting if a user is vocalizing is to measure and track the acoustic energy picked up by the microphone by calculating the root mean square (RMS) of the signal over a time window [5].

F_0 Estimation

The F_0 estimator uses an autocorrelation method[130], which is simple and computationally efficient. The autocorrelation function $r_t(\tau)$, (See Eq. F.1) uses short time windows of length W_{auto}; where if the signal is periodic then $r_t(\tau)$ will have peaks at lags τ corresponding to the signal's period. To estimate F_0 we find the time lag corresponding to the first peak in the autocorrelation function. For further details on the algorithm refer to our previous publication [5].

$$r_t(\tau) = \sum_{(j=t+1)}^{(t+W_{auto})} [x_j x_{j+\tau}(1 - \tau/\tau_{max})] \tag{F.1}$$

The sampling rate of the sensor, F_s, was set at 8 kHz. We used sinusoidal test signals to determine an estimation frequency range of 120–800 Hz. This

range covers the frequencies we would expect from children's and most adults' vocalizations.

F_0 Correction

The F_0 correction helps us to distinguish sounds emitted from the environment. For filtering outlier we define an interquartile range IQR as $Q_3 + 1.5 \times IQR$ and $Q_1 - 1.5 \times IQR$, where any F_0 estimate inside this boundaries is considered voice. For the rest of the cases we make an octave correction. If the difference between its two neighbour estimates is larger than 10 Hz, we consider the outlier as noise.

When the F_0 correction is not used in the voice sensor, the mean error can be as high as 14 Hz as compared to when its working it drops down to 4 Hz.

User tests and functional demonstrations

We performed several tests with the current version of Fable which include: 1) a user test with children to validate the usability of the connectors, 2) a snake and 3) a quadruped robot to validate the systems ability to build functional mobile robots, 4) a two programming workshops with teenagers and 5) the use of voice control application by kids in an after-school club.

Connector

As we mentioned in Section F.2, children should be able to attach and detach the Fable modules, therefore we performed a user test at an after school club in the Copenhagen area. At the time of the test we lacked finished prototypes, so instead we used solid 100 mm tall PVC cylinders with fully assembled connectors attached to one end. Both the medium and small sized connectors, prototyped with SLS technology, were tested. The diameters of the cylinders were 40 mm for the small connector and 60 mm for the medium. Since the target age group for Fable is 6 and up, we selected a group of children between 6-10 years. Where 9 out of the group were 6, 1 was 7, 3 were 8, 2 were 9 and 1 was 10 years old.

The children were asked to connect and disconnect the PVC modules several times while we observed if they encountered any difficulty. Most of the children did not have any issues connecting the modules. Four kids, mainly 6 years old, had little difficulties connecting during the first couple of tries until they figured out how to connect them. Sometimes it was sufficient for them to imitate the process, by watching an older child succeed. We also observed that the force required for disassembly exceeded that of most 6 year olds. Further tests with different

connector prototypes are needed to determine an appropriate design that enables children to comfortably detach modules.

Locomotion

After developing our first set of modules we decided to test the locomotion capabilities of the system by arranging the modules in two configurations: as a quadruped and as a snake. The configurations that we could achieve were limited due to the amount of modules available, which were four 2DOF Joint Modules and one 4-Way Branch.

Quadruped locomotion. We built an 8 DoF quadruped by connecting a 2 DoF Joint module to each end of a 4 Way Branch. The configuration weighs a total of 2024 g, including battery and controller.

We implemented Central Pattern Generators (CPG) based on the architectures used in previous work [3, 110]. We implemented four gaits in the design: forward, backward and clockwise and counter-clockwise turns. To control the robot we used the analog stick of a gamepad connected to a PC. The computer had a wireless Zigbee communication, that only sent a high level control signal to the robot. All CPG computation was performed locally by the embedded microcontroller. To study the robot's mobility the test setup was based on two types of floors with different friction coefficients: a foam based mat and a linoleum based floor. We measured the walking speed by attaching a string to the robots body and measuring the time it took it to stretch the string to a distance of 1.5 m. The turn speed was made by averaging the time it took the robot to make 10 full revolutions. On foam, the average turn rate was 0.18 rev/s while on Linoleum was 0.15 rev/s. We observed that the quadruped moved with a reasonable speed and turning rate compared to its size and that the CPG architecture produced natural smooth transitions when switching between gaits. Further, we encountered no issues with servos being too weak or connectors disconnecting unintentionally.

Snake locomotion. Afterwards we also built a snake by snapping four 2 DoF JMs in series, which weighed a total of 1771 g, including the battery and electronics. We then implemented a side-winding gait based on the same CPG architecture as for the quadruped. The frequency of the gait was controlled using one of the gamepad's analog stick.

It was difficult to measure more due to the fact that the mat used was relatively small and since the snakes movement was not straight it tended to fall of the mat at longer distances. The distance was measured with two strips that marked the 60 cm. Ten attempts were registered in total for each scenario. The average speed

for the Foam mat scenario is of 0.17 m/s and for the Linoleum is of 0.19 m/s. We observed that although the snake velocity is slightly higher than the quadruped velocity the snake is much less controllable (not moving straight, flipping over). Moreover, we found no efficient gaits for turning or forward locomotion. To address these issues we will consider including the possibility of adding passive wheels to the Joint modules so that more effective gaits can be implemented.

Pico LOGO

In Section F.2 we mentioned that Fable uses a programming language called Pico LOGO. Even though Pico LOGO is considered an easy to use programming language we were not sure it was a suitable language for users without programming experience. We performed two tests at the Mærsk Mckinney Møller Institute at the municipality of Sorø. Each test involved a programming task, the first one made use of the vision sensor and followed a tutorial approach, while the second one was a more open approach that consisted on the development of a walking gate for the 8 DoF quadruped.

Vision Sensor. After developing an interface for the Asus vision sensor, we wanted to test and see if inexperienced users were able to program a humanoid torso that would play "simon says" by mimicking the position of their shoulders and elbows. For this application we developed a step by step tutorial that explained the task for the robot that played simon says by leading the game. Once the students had accomplished the tutorial they were asked to develop the code for the robot to mimic the user. The workshop consisted of 13 teenagers divided in 5 groups of 2 and one of 3. Each group was given a laptop and a USB flash memory used to feed the applications to the system. After close to two hours, most of the teams were able to finish the tutorial, Fig. F.1 represents a code developed by following the tutorial for this workshop.

Walking gait. The workshop consisted of 10 teenagers divided in groups of 2. Each group had a laptop and a USB flash memory used to feed the applications to the system. The task consisted of developing a walking gate for a quadruped robot. At the beginning of the workshop we gave a 10 minute introduction to the programming environment. The workshop ran for three hours where the users were allowed to experiment with their code and download the program to the robot by using their USB memory stick. None of the teams had issues reprogramming the robot. By the end of the workshop none of the teams managed to program a full walking gate, although the majority of the students showed signs of enthusiasm and were excited about their experience with Fable, and some were even disappointed

that the workshop had to end.

Listing F.1: Example of code obtained by following the tutorial given at the programming workshop.

```
constants [
  [userId 1]
  [left_elbow 0]
  [motorId 1]
]

to mirror_elbow
  ; Define a variable
  let [angle 0]

  ; Assigns the variable with elbow angle
  make "angle get_joint_angle userId left_elbow

  ; Set the motor position to angle
  set_motor_pos motorId :angle

  wait 1
end

to onstart
  ; Requests for left elbow data
  use_user_joint left_elbow

  ; Infinite loop
  forever [mirror_elbow]
end
```

Voice control

Once we developed the voice control application we wanted to know if kids would find it amusing to interact with a robot by controlling the pitch of their voice. To test this we gathered a group of 8 children ages 7 to 10 and had them interact with the robot in two ways. As the average F_0 can vary across age and gender we calculated the individual parameters $F_{0,min}$ and $F_{0,max}$ prior to the tests. The first test involved the control of a humanoid torso while the second one was to control a walking robot.

Humanoid Torso The first application was used as an introduction to how the users could voice control the robot. The game's goal was to match one of Fable's arm angle to match the position of the other arm. The target position in one arm was initially set with a gamepad by the experimenter. The other arm was controlled by the F_0 of the child's voice. The child hummed or sung a tone into the microphone and the arm would raise or lower its position based on the F_0 that was produced.

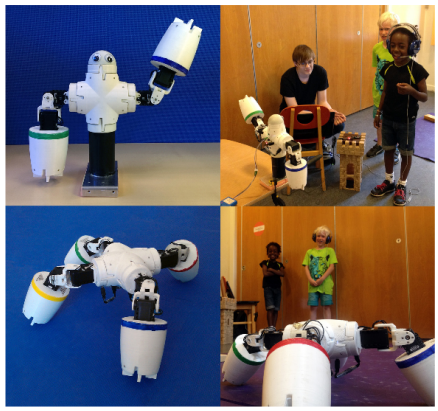


Figure F.6: Starting clockwise from upper-left corner. A humanoid torso built with four Fable modules. A child controlling a humanoid torso through his voice. A child controlling a quadruped through voice control. A Quadruped made of 5 Fable modules.

Once the controlled arm matched the target, the Fable torso waved its arms and it would continue for as long as the user could hold the tone. We consider this game as a good introduction since users receive a clear visual feedback from the robot corresponding to the pitch of their voice. Three random target values were used for each child and all of the children were able to finish the game.

Initially, two of the younger children had difficulties because they were changing their pitch too fast and/or in large steps. As well, the younger girls required more instruction before understanding the relationship between their voice and Fable's arm position. Overall the children were very engaged in the activity and seemed to have fun playing with Fable.

Walking Robot In the following test the objective of the game was to move the quadruped robot through a track with four gates. The F_0 of the users' voices was used to select between four different movements: forward, backward, left-turn, right-turn. We also provided visual feedback on a computer screen, where we plotted the F_0 being produced as a large dot that moved over a background. The background, illustrated the target pitch corresponding to each of the 4 motions.

Single Player: In this game, there were two different levels of difficulties. At level 1, there were only two pitch targets corresponding to forward and left turn movements. At level 2, there were four pitch targets, each corresponding to one of the four movements (see Fig. F.6).

Two Player - Co-op: In this game, two players collaborated to control the quadruped using two pitch targets. One user controls forward and backward

movement while the other controls turning left and right.

Two Player - Vs: In this game, two players competed to gain control over the quadruped and move it to a specific area to win the game. One user controls forward and left-turn and the other controls backward and right-turn. The user that was loudest, while maintaining a target pitch, controlled the robot.

All eight children participated in the single player game at level 1. The four oldest quickly understood the task and learned how to shift between tones to get the robot moving through the track. In contrast, only one of the younger boys performed well. The two oldest boys tried level 2 and were able to control the robot successfully. Overall the kids seemed to enjoy the gameplay in collaborative more than in any of the previous ones.

F.4 Lessons Learned: Designing Fable Version 2

Not all of the design aspects of Fable work well, therefore we have planned to make improvements in some of the system's key features. We are working on the development of Fable that does not include electronics on every module. We are currently working on approach that uses RF communication with the PC, because neighbor to neighbor communication adds plenty of cost as well as complexity to the systems software architecture, while only bringing as an advantage the awareness of topology. On the other hand wireless communication on some modules will allow us to develop passive modules, that is modules without any electronics, it will bring down the cost of the system but rely on the user to know the topology.

We also need to develop more types of modules: a wheel module, sensor modules, many other branching modules, small sized modules as well as termination modules.

Regarding the connector design, we need to improve the mechanical interface by reducing the size of the flanges and putting the magnets on the back side to avoid magnets from ripping themselves apart from the connectors.

Even though Pico LOGO is a good, minimalistic and yet powerful language, it seems a bit abstract for users without any programming experience [153]. We intend to experiment with platforms similar to MIT MediaLab's Scratch programming [137] or Google's blockly. In this way we can have young children program the system in an intuitive manner.

F.5 Conclusion

Throughout this paper we have described the development of a modular robotic playware platform called Fable. We presented the connector design as well as a set of two types of modules: Branching Modules and Termination Modules. Furthermore we described the system's electronics as well as the functionality of the USB port presented in the 4 Way Branch. We then continued to present the software architecture and then followed to describe some example applications using a 3D vision sensor and a voice sensor. Then we talked about testing the system in several ways which include: the connectors' functionality, locomotion, programming the system with the vision sensor and playing games with the voice sensor.

There is still a long list of improvements that we intend to implement in further iterations of the system. We expect that by sharing our experience in the development of Fable will help bridge the gap between modular robotic systems and consumer electronics.

Acknowledgment

Part of this work was developed under the "Modular Playware Technology" project and funded by the Danish National Advanced Technology Foundation. We would like to thank Arnthor Magnusson, Bjarke Heesche, Rune Fogh, Brian Silverman, Mikael Moghadam and Ewen MacDonald for their collaboration.

Bibliography

[1] P. Goodman. *Compulsory miseducation*. Penguin Harmondsworth, 1971.

[2] I. Illich. "Deschooling society". *New York* 56 (1971).

[3] M. Pacheco, M. Moghadam, A. Magnússon, B. Silverman, H. H. Lund, and D. J. Christensen. "Fable: Design of a modular robotic playware platform". *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 544–550.

[4] A. Magnússon, M. Pacheco, M. Moghadam, H. H. Lund, and D. J. Christensen. "Fable: Socially interactive modular robot". *The Eighteenth International Symposium on Artificial Life and Robotics 2013*. 2013.

[5] B. Heesche, E. MacDonald, R. Fogh, M. Pacheco, and D. J. Christensen. "Playful interaction with voice sensing modular robots". *Social Robotics*. IEEE. 2013, pp. 180–189.

[6] M. Pacheco, R. Fogh, H. H. Lund, and D. J. Christensen. "Fable: A Modular Robot for Students, Makers and Researchers". *Proceedings of the IROS workshop on Modular and Swarm Systems: from Nature to Robotics*. 2014.

[7] M. Pacheco, R. Fogh, H. Lund, and D. Christensen. "Fable II: Design of a Modular Robot for Creative Learning". *Proceedings of 2015 IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 6134–6139.

[8] K. Peter. *Computing in the national curriculum: A guide for secondary teachers*. National Association of Advisors for Computers in Education, 2014.

[9] E. Giannisakis and H. H. Lund. "Combining playware exergaming with a mobile fitness app". *Journal of Robotics, Networks and Artificial Life* 1.4 (2015), pp. 249–252.

[10] H. H. Lund. "Playware Research–Methodological Considerations". *Journal of Robotics, Networks and Artificial Life* 1.1 (2014), pp. 23–27.

[11] H. H. Lund and C. Jessen. “Playware–intelligent technology for children’s play” (2005).

[12] L. Pagliarini and H. H. Lund. “Integrating modular mechatronic systems for immersive performances”. *Advanced Intelligent Mechatronics (AIM), 2015 IEEE International Conference on*. IEEE. 2015, pp. 353–358.

[13] T. D. M. of Education. *Improving the Public School*. The Danish Ministry of Education, 2014.

[14] E. Danish Ministry for Children and G. Equality. *The Folkeskole - Ministry of Education*. `http://eng.uvm.dk/Education/Primary-and-lower-secondary-education/The-Folkeskole`. Accessed: 2016-03-25.

[15] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. “Modular self-reconfigurable robot systems [grand challenges of robotics]”. *Robotics & Automation Magazine, IEEE* 14.1 (2007), pp. 43–52.

[16] T. Fukuda and Y. Kawauchi. “Cellular robotic system (CEBOT) as one of the realization of self-organizing intelligent universal manipulator”. *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE. 1990, pp. 662–667.

[17] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss. “Structure decision method for self organising robots based on cell structures-CEBOT”. *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. IEEE. 1989, pp. 695–700.

[18] J. W. Romanishin, K. Gilpin, and D. Rus. “M-blocks: Momentum-driven, magnetic modular robots”. *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 4288–4295.

[19] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund. “Design of the ATRON Lattice-Based Self-Reconfigurable Robot”. *Autonomous Robots* 21 (2006), pp. 165–183.

[20] S. Murata and H. Kurokawa. “Self-reconfigurable robots”. *Robotics & Automation Magazine, IEEE* 14.1 (2007), pp. 71–78.

[21] A. Sproewitz, P. Laprade, S. Bonardi, M. Mayer, R. Moeckel, P.-A. Mudry, and A. J. Ijspeert. “Roombots—Towards decentralized reconfiguration with self-reconfiguring modular robotic metamodules”. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 1126–1132.

[22] M. Fujita, H. Kitano, and K. Kageyama. “A reconfigurable robot platform”. *Robotics and Autonomous Systems* 29.2 (1999), pp. 119–132.

[23] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson. “Robotics: Self-reproducing machines”. *Nature* 435.7039 (2005), pp. 163–164.

[24] K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, and S. Kokaji. “Self-assembly and self-repair method for a distributed mechanical system”. *Robotics and Automation, IEEE Transactions on* 15.6 (1999), pp. 1035–1045.

[25] S. Murata, H. Kurokawa, and S. Kokaji. “Self-assembling machine”. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE. 1994, pp. 441–448.

[26] H. G. Marques, M. Christophe, A. Lenz, K. Dalamagkidis, U. Culha, M. Siee, P. Bremner, and the MYOROBOTICS Project Team. “MYOROBOTICS: a modular toolkit for legged locomotion research using musculoskeletal designs”. *Proc. 6th International Symposium on Adaptive Motion of Animals and Machines (AMAM’13)*. Darmstadt, Germany, 2013.

[27] J. C. Larsen and K. Støy. “LocoKit-A Construction Kit for Exploration of Morphology of Legged Robots” (2011).

[28] P. Jantapremjit and D. Austin. “Design of a modular self-reconfigurable robot”. *Proceedings of Australian conference on robotics and automation*. 2001, pp. 38–43.

[29] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson. “Evolved and designed self-reproducing modular robotics”. *Robotics, IEEE Transactions on* 23.2 (2007), pp. 308–319.

[30] A. Sproewitz, A. Billard, P. Dillenbourg, and A. Ijspeert. “Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture”. *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE. 2009, pp. 4259–4264.

[31] K. Kotay, D. Rus, M. Vona, and C. McGray. “The self-reconfiguring robotic molecule: Design and control algorithms”. *Workshop on Algorithmic Foundations of Robotics*. Citeseer. 1998, pp. 376–386.

[32] C. M. D. Rus and N. Hanover. “Self-reconfigurable molecule robots as 3d metamorphic robots” (1998).

[33] M. Yim, D. G. Duff, and K. D. Roufas. “PolyBot: a modular reconfigurable robot”. *Robotics and Automation, 2000. Proceedings. ICRA’00. IEEE International Conference on*. Vol. 1. IEEE. 2000, pp. 514–520.

[34] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw. "Connecting and disconnecting for chain self-reconfiguration with PolyBot". *Mechatronics, IEEE/ASME Transactions on* 7.4 (2002), pp. 442–451.

[35] M. Yim. "A reconfigurable modular robot with many modes of locomotion". *Proc. of Intl. Conf. on Advanced Mechatronics*. Tokyo, Japan. 1993, pp. 283–288.

[36] A. Castano, A. Behar, and P. M. Will. "The Conro modules for reconfigurable robots". *Mechatronics, IEEE/ASME Transactions on* 7.4 (2002), pp. 403–409.

[37] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. "M-TRAN: self-reconfigurable modular robotic system". *Mechatronics, IEEE/ASME Transactions on* 7.4 (Dec. 2002), pp. 431–441.

[38] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata. "Distributed self-reconfiguration of M-TRAN III modular robotic system". *The International Journal of Robotics Research* 27.3-4 (2008), pp. 373–386.

[39] G. Studer and H. Lipson. "Spontaneous emergence of self-replicating, competing cube species in physical cube automata". *GECCO Late Breaking Paper, Proceedings, Washington DC, USA* (2005).

[40] V. Zykov, A. Chan, and H. Lipson. "Molecubes: An open-source modular robotics kit". *IROS-2007 Self-Reconfigurable Robotics Workshop*. Citeseer. 2007, pp. 3–6.

[41] V. Zykov, W. Phelps, N. Lassabe, and H. Lipson. "Molecubes extended: Diversifying capabilities of open-source modular robotics". *IROS-2008 Self-Reconfigurable Robotics Workshop*. Citeseer. 2008.

[42] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, and E. Klavins. "Modular Self-reconfigurable Robot Systems: Challenges and Opportunities for the Future". *IEEE Robotics & Automation Magazine* 14.1 (Mar. 2007), pp. 43–52.

[43] B. Salemi, M. Moll, and W. Shen. "SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system". *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 3636–3641.

[44] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu. "Miche: Modular shape formation by self-disassembly". *The International Journal of Robotics Research* 27.3-4 (2008), pp. 345–372.

[45] K. C. Cheung, E. D. Demaine, J. R. Bachrach, and S. Griffith. "Programmable assembly with universally foldable strings (moteins)". *Robotics, IEEE Transactions on* 27.4 (2011), pp. 718–729.

[46] M. Rubenstein, A. Cornejo, and R. Nagpal. "Programmable self-assembly in a thousand-robot swarm". *Science* 345.6198 (2014), pp. 795–799.

[47] C. D. Onal, R. J. Wood, and D. Rus. "Towards printable robotics: Origami-inspired planar fabrication of three-dimensional mechanisms". *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 4608–4613.

[48] E. Hawkes, B. An, N. Benbernou, H. Tanaka, S. Kim, E. Demaine, D. Rus, and R. Wood. "Programmable matter by folding". *Proceedings of the National Academy of Sciences* 107.28 (2010), pp. 12441–12445.

[49] Modular Robotics LLC. *Cubelets Robot Construction for Kids*. Accessed: 2015-11-25. Dec. 2011.

[50] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. "Robot programming by demonstration". In: *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[51] Arthur Sacek. *LEGO RCX Lathe*. Accessed: 2015-11-25. Dec. 2011.

[52] S. Papert. *Mindstorms: children, computers, and powerful ideas*. New York, NY, USA: Basic Books, Inc., 1980.

[53] F. Klassner and S. D. Anderson. "Lego MindStorms: Not just for K-12 anymore". *IEEE Robotics & Automation Magazine* 10.2 (2003), pp. 12–18.

[54] J. Nielsen, N. K. Brendsen, and C. Jessen. "RoboMusicKids–music education with robotic building blocks". *Digital Games and Intelligent Toys Based Education, 2008 Second IEEE International Conference on*. IEEE. 2008, pp. 149–156.

[55] H. H. Lund. "Intelligent artefacts". *Proceedings of the 8 th International symposium on artificial life and robotics, ISAROB*. Citeseer. 2003, pp. I11–I14.

[56] H. H. Lund. "Modular Playware Technology: A Brief Historical Review". *2012 Seventeenth International Symposium on Artificial Life and Robotics*. 2012.

[57] H. S. Raffle, A. Parkes, and H. Ishii. "Topobo: a constructive assembly system with kinetic memory". *Proceedings of the SIGCHI conference on Human factors in computing systems*. CHI '04. Vienna, Austria: ACM, 2004, pp. 647–654.

[58] LEGO. *1039-1: Manual Control Set 1 | Brickset: LEGO set guide and database*. http://brickset.com/sets/1039-1/Manual-Control-Set-1. Accessed: 2016-06-10.

[59] K. Stoy, A. Lyder, R. F. M. Garcia, and D. J. Christensen. "Hierarchical robots". *IROS Workshop on Self-Reconfigurable Modular Robots*. 2007.

[60] E. Schweikardt and M. D. Gross. "roBlocks: a robotic construction kit for mathematics and science education". *Proceedings of the 8th international conference on Multimodal interfaces*. ICMI '06. Banff, Alberta, Canada: ACM, 2006, pp. 72–75.

[61] E. Schweikardt and M. D. Gross. "The robot is the program: interacting with roBlocks". *Proceedings of the 2nd international conference on Tangible and embedded interaction*. ACM. 2008, pp. 167–168.

[62] K. Stoy, D. Brandt, and D. J. Christensen. *Self-Reconfigurable Robots: An Introduction*. Intelligent Robotics and Autonomous Agents series. The MIT Press, 2010.

[63] J. Piaget. "Science of education and the psychology of the child. Trans. D. Coltman." (1970).

[64] I. E. Harel and S. E. Papert. *Constructionism*. Ablex Publishing, 1991.

[65] LEGO. *Home - LEGO MINDSTORMS*. http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com. Accessed: 2016-06-10.

[66] VEX. *VEX EDR - VEX Robotics*. http://www.vexrobotics.com/vexedr/. Accessed: 2016-06-10.

[67] SoftBank Robotics. *SoftBank Robotics | Humanoid robotics & programmable robots*. http://www.vexrobotics.com/vexedr/. Accessed: 2016-06-10.

[68] J. Kershner. "Aldebaran's NAO robot educates kids with autism" (2011).

[69] Playful Invention Company. *Playful Invention Company*. http://http://www.playfulinvention.com/. Accessed: 2016-06-10.

[70] iRobot. *iRobot - Create2*. http://www.irobot.com/About-iRobot/STEM/Create-2.aspx. Accessed: 2016-06-10.

[71] Innvo Labs. *PLEOworld*. http://www.pleoworld.com/pleo_rb/eng/index.php. Accessed: 2016-06-10.

[72] Thymio. *Thymio - Thymio & Aseba*. https://www.thymio.org/en:thymio. Accessed: 2016-06-10.

[73] LEGO. *LEGO.com Primary School - WeDO 2.0*. https://education.lego.com/en-gb/lesi/elementary/wedo-2. Accessed: 2016-06-10.

[74] ROBOTIS. *:::ROBOTIS:::* http://www.robotis.com/xe/ollo_en. Accessed: 2016-06-10.

[75] Sphero. *Sphero | Connected Toys*. http://www.sphero.com/. Accessed: 2016-06-10.

[76] Eva P. Christensen is an Educational Consultant from University Colleage Zealand. *Interviewed:* 2015-11-10.

[77] Robokind. *Zeno-R25-*. http://www.robokindrobots.com/zeno-r25/. Accessed: 2016-06-10.

[78] S. Costa, F. Soares, and C. Santos. "Facial expressions and gestures to convey emotions with a humanoid robot". In: *Social Robotics*. Springer, 2013, pp. 542–551.

[79] Barobo Inc. *Barobo, Inc. | Educational Robotics*. https://www.barobo.com/. Accessed: 2016-06-10.

[80] Sphero. *Ollie | Connected Toy for iOS and Android*. http://www.sphero.com/ollie. Accessed: 2016-06-10.

[81] Wonder Workshop. *Wonder Workshop | Meet Dash*. https://www.makewonder.com/dash. Accessed: 2016-06-10.

[82] Origami Robotics. *Origami Robotics*. http://www.origamirobotics.com/. Accessed: 2016-06-10.

[83] Makeblock. *mBot-Blue (Bluetooth Version) | Makeblock*. http://makeblock.com/mbot-stem-educational-robot-kit-for-kids/. Accessed: 2016-06-10.

[84] M. Rubenstein, B. Cimino, R. Nagpal, and J. Werfel. "AERobot: An affordable one-robot-per-student system for early robotics education". *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 6107–6113.

[85] Ozobot. *Ozobot | Smart Toy Robot*. http://ozobot.com/. Accessed: 2016-06-10.

[86] The Local. *Denmark's public schools enter a new era*. Accessed: 2016-04-11.

[87] E. C. Jakob Wandall. *Lockout and reform: A turbulent year for schools in Denmark*. Accessed: 2016-04-11.

[88] ASTM. “Standard Guide for Rapid Prototyping of Information Systems” (2010).

[89] S. Mueller, B. Kruck, and P. Baudisch. “LaserOrigami: laser-cutting 3D objects”. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, pp. 2585–2592.

[90] Google. *blockly - A visual programming editor*. `https://code.google.com/p/blockly/`. Accessed: 2014-06-26.

[91] L. Righetti and A. J. Ijspeert. “Pattern generators with sensory feedback for the control of quadruped locomotion”. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 819–824.

[92] M. D. Gross and M. Eisenberg. “Why Toys Shouldn’t Work" Like Magic": Children’s Technology and the Values of Construction and Control”. *Digital Game and Intelligent Toy Enhanced Learning, 2007. DIGITEL’07. The First IEEE International Workshop on*. IEEE. 2007, pp. 25–32.

[93] Shape Robotics. *Shape Robotics*. Accessed: 2015-11-28. Nov. 2015.

[94] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata. “Distributed Self-Reconfiguration of M-TRAN III Modular Robotic System”. *International Journal of Robotics Research* 27.3-4 (2008), pp. 373–386.

[95] M. Yim, B. Shirmohammadi, and D. Benelli. “Amphibious Modular Robot Astrobiologist”. *Proceedings of SPIE Unmanned Systems Technology IX*. Vol. 656. Apr. 2007.

[96] J. C. Larsen, R. F. M. Garcia, and K. Stoy. “Increased Versatility of Modular Robots through Layered Heterogeneity”. *Proceedings of the ICRA Workshop on Modular Robots, State of the Art*. Anchorage, Alaska, May 2010, pp. 24–29.

[97] A. Sproewitz, S. Pouya, S. Bonardi, J. van den Kieboom, R. Mockel, A. Billard, P. Dillenbourg, and A. Ijspeert. “Roombots: Reconfigurable Robots for Adaptive Furniture”. *IEEE Computational Intelligence Magazine, special issue on Evolutionary and developmental approaches to robotics* 5.3 (2010), pp. 20–32.

[98] H. H. Lund, N. K. Bærendsen, J. Nielsen, C. Jessen, and K. Falkenberg. “RoboMusic with Modular Playware”. *Proceedings of International Symposium on Artificial Life and Robotics (AROB’10)*. 2010.

[99] H. Lund. “Anybody, Anywhere, Anytime - Robotics with a Social Impact through a Building Block Approach”. *Proceedings of IEEE Workshop of Advanced Robotics and its Social Impact (ARSO)*. CA, United States, 2011.

[100] H. Lund and P. Marti. “Designing modular robotic playware”. *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*. IEEE. 2009, pp. 115–121.

[101] J. Saldien, K. Goris, S. Yilmazyildiz, W. Verhelst, and D. Lefeber. “On the design of the huggable robot Probo”. *Journal of Physical Agents* 2.2 (2008), pp. 3–12.

[102] W. D. Stiehl, J. K. Lee, C. Breazeal, M. Nalin, A. Morandi, and A. Sanna. “The huggable: a platform for research in robotic companions for pediatric care”. *Proceedings of the 8th International Conference on interaction Design and Children*. ACM. 2009, pp. 317–320.

[103] C. Breazeal and B. Scassellati. “A context-dependent attention system for a social robot”. *IJCAI-99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Vols 1 & 2* (1999), pp. 1146–1151.

[104] E. H. Østergaard, K. Kassow, R. Beck, and H. Lund. “Design of the ATRON lattice-based self-reconfigurable robot”. *Autonomous Robots* 21.2 (2006), pp. 165–183.

[105] J. Sastra, S. Chitta, and M. Yim. “Dynamic Rolling for a Modular Loop Robot”. *Proc. of Intl. Symp. on Experimental Robotics*. Rio de Janeiro Brazil, 2006.

[106] E. 71-1:2005+A8:2009. *Safety of toys - Part 1: Mechanical and physical properties*. CEN, Brussels, Belgium.

[107] D. J. Christensen, U. P. Schultz, and M. Moghadam. “The Assemble and Animate Control Framework for Modular Reconfigurable Robots”. *Proceedings of the IROS Workshop on Reconfigurable Modular Robotics: Challenges of Mechatronic and Bio-Chemo-Hybrid Systems*. 2011.

[108] P. invention company. *http://www.picocricket.com/picopeople.html*. Ed. by C. Ltd. Developers of PicoLOGO.

[109] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. “ROS: an open-source Robot Operating System”. *ICRA Workshop on Open Source Software*. 2009.

[110] D. J. Christensen, A. Sproewitz, and A. J. Ijspeert. “Distributed Online Learning of Central Pattern Generators in Modular Robots”. *Proceedings of the 11th International Conference on Simulation of Adaptive Behavior (SAB2010)*. Paris, France, Aug. 2010, pp. 402–412.

[111] T. Fukuda and S. Nakagawa. “Dynamically reconfigurable robotic system”. *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA'88)*. 1988, pp. 1581–1586.

[112] K. W. Gilpin, A. N. Knaian, and D. L. Rus. “Robot pebbles: One centimeter modules for programmable matter through self-disassembly”. *IEEE* (2010).

[113] A. Lyder, R. F. M. Garcia, and K. Stoy. “Mechanical Design of Odin, an Extendable Heterogeneous Deformable Modular Robot”. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2008)*. Nice, France, Sept. 2008, pp. 883–888.

[114] P. Marti, L. Giusti, and H. H. Lund. “The role of modular robotics in mediating nonverbal social exchanges”. *IEEE Transactions on Robotics* 25.3 (June 2009), pp. 602–613.

[115] C. B. Nielsen and H. H. Lund. “Adapting Playware to Rehabilitation Practices”. *Serious Games - Theory, Technology & Practice: Proceedings GameDays 2011*. 2011.

[116] C. L. Breazeal. *Designing sociable robots*. The MIT Press, 2004.

[117] A. J. N. van Breemen. “Bringing Robots To Life: Applying Principles Of Animation To Robots”. *Proceedings of the CHI2004 Workshop on Shaping Human-Robot Interaction - Understanding the Social Aspects of Intelligent Robotic Products*. Vienna, 2008.

[118] K. Goris, J. Saldien, I. Vanderniepen, and D. Lefeber. “The huggable robot Probo, a multi-disciplinary research platform”. *Communications in Computer and Information Science* 33 CCIS (2009), pp. 29–41.

[119] S. S. O. Adalgeirsson and C. Breazeal. “MeBot: A robotic platform for socially embodied presence”. *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction (HRI '10)*. 2010, pp. 15–22.

[120] H. H. Lund and T. Thorsteinsson. “Social playware for mediating tele-play interaction over distance”. English. *Artificial Life and Robotics* 16 (4 2012), pp. 435–440.

[121] E. Rosten and T. Drummond. “Fusing points and lines for high performance tracking.” *IEEE International Conference on Computer Vision*. Vol. 2. Oct. 2005, pp. 1508–1511.

[122] D. G. Lowe. “Object Recognition from Local Scale-Invariant Features”. *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV ’99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–.

[123] M. Muja and D. G. Lowe. “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration”. *International Conference on Computer Vision Theory and Application VISSAPP’09)*. INSTICC Press, 2009, pp. 331–340.

[124] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, and C. J. Taylor. “Towards robotic self-reassembly after explosion”. *Departmental Papers (MEAM)* (2007), p. 147.

[125] H. H. Lund. “Modular interactive tiles for rehabilitation: evidence and effect”. *Proceedings of the 10th WSEAS international conference on Applied computer science*. World Scientific, Engineering Academy, and Society (WSEAS). 2010, pp. 520–525.

[126] M. Resnick and B. Silverman. “Some reflections on designing construction kits for kids”. *Proceedings of the 2005 conference on Interaction design and children*. ACM. 2005, pp. 117–122.

[127] M. Resnick. “All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten”. *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*. ACM. 2007, pp. 1–6.

[128] R. P. Lippmann. “Speech recognition by machines and humans”. *Speech communication* 22.1 (1997), pp. 1–15.

[129] C. Breazeal and L. Aryananda. “Recognition of affective communicative intent in robot-directed speech”. *Autonomous robots* 12.1 (2002), pp. 83–104.

[130] A. De Cheveigné and H. Kawahara. “YIN, a fundamental frequency estimator for speech and music”. *The Journal of the Acoustical Society of America* 111.4 (2002), pp. 1917–1930.

[131] F. Jacobsen, T. Poulsen, J. H. Rindel, A. C. Gade, and M. Ohlrich. “Fundamentals of acoustics and noise control”. *Technical University of Denmark, Department of Electrical Engineering, Note* 31200 (2011).

[132] S. Steidl. *Automatic classification of emotion related user states in spontaneous children's speech*. University of Erlangen-Nuremberg Germany, 2009.

[133] P. Boersma and D. Weenick. *Computer Software*. University of Amsterdam, Amsterdam, 2006.

[134] ModularRobotics. *MOSS Robot Construction System*. `http://www.modrobotics.com/moss/`. Accessed: 2014-06-26.

[135] M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, T. Selker, and M. Eisenberg. "Design principles for tools to support creative thinking". *NSF Workshop on Report on Creativity Support Tools*. 2005.

[136] M. Csikszentmihalyi. *Flow: The psychology of optimal performance*. 1990.

[137] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. "Scratch: programming for all". *Communications of the ACM* 52.11 (2009), pp. 60–67.

[138] E. Hippel. "User toolkits for innovation". *Journal of product innovation management* 18.4 (2001), pp. 247–257.

[139] D. W. Dahl and C. P. Moreau. "Thinking inside the box: Why consumers enjoy constrained creative experiences". *Journal of Marketing Research* 44.3 (2007), pp. 357–369.

[140] S. L. Brown. *Play: How it shapes the brain, opens the imagination, and invigorates the soul*. Penguin, 2009.

[141] L. P. Rieber. "Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games". *Educational technology research and development* 44.2 (1996), pp. 43–58.

[142] R. Fogh and A. Johansen. "The play grid". *Proceedings of the 12th International Conference on Interaction Design and Children - IDC '13* (2013), pp. 356–359.

[143] Y. B. Kafai and M. Resnick. *Constructionism in practice: Designing, thinking, and learning in a digital world*. Routledge, 1996.

[144] D. J. Christensen, U. P. Schultz, and K. Stoy. "A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots". *Robotics and Autonomous Systems* 61.9 (2013), pp. 1021–1035.

[145] W. Shen, P. Will, A. Galstyan, and C. Chuong. "Hormone-inspired self-organization and distributed control of robotic swarms". *Autonomous Robots* 17.1 (2004), pp. 93–105.

[146] R. Lal and R. Fitch. “A Hardware-in-the-Loop Simulator for Distributed Robotics”. *Proceedings of ARAA Australasian Conference on Robotics and Automation (ACRA)*. 2009, pp. 544–550.

[147] J. Davey, J. Sastra, M. Piccoli, and M. Yim. “ModLock: A manual connector for reconfigurable modular robots”. *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. Oct. 2012, pp. 3217–3222.

[148] *LEGO Education Mindstorms EV3 Building Guide*. 2014.

[149] M. U. Bers, I. Ponte, C. Juelich, A. Viera, and J. Schenker. “Teachers as designers: Integrating robotics in early childhood education”. *Information Technology in Childhood Education Annual* 2002.1 (2002), pp. 123–145.

[150] F. Martin, B. Mikhak, and B. Silverman. “MetaCricket: A designer’s kit for making computational devices”. *IBM Systems Journal* 39.3.4 (2000), pp. 795–815.

[151] J. Fast. *Body language*. Simon and Schuster, 1970.

[152] E. Rosten and T. Drummond. “Machine learning for high-speed corner detection”. In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[153] C. M. Lewis. “How programming environment shapes perception, learning and goals: logo vs. scratch”. *Proceedings of the 41st ACM technical symposium on Computer science education*. SIGCSE ’10. Milwaukee, Wisconsin, USA: ACM, 2010, pp. 346–350.

Technical University of Denmark
Automation and Control (AUT)
Elektrovej Building 326
DK-2800, Kgs. Lyngby
Denmark
Phone: (+45) 45 25 35 76
Email: info@elektro.dtu.dk
www.elektro.dtu.dk

ISBN: 978-87-92465-61-0