Aalto-yliopisto
Insinööritieteiden
korkeakoulu

Jani Kalasniemi

Capturing Participant Data from Product Design Process

Triangulation of Three Different Approaches

Master's thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in Technology

Espoo, May 2nd, 2016

Supervisor:	Professor Kalevi Ekman
Thesis advisors:	Lauri Repokari, Lic.Sc (Tech.)
	Joona Kurikka, M.Sc. (Tech.)

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO

www.aalto.fi

Abstract of master's thesis

Author Jani Kalasniemi	
Title of thesis Capturing Participant Data from Product Design Process – Triangulation of Three Different Approaches	
Degree programme Mechanical Engineering	
Major/minor Machine Design	**Code** K3001
Thesis supervisor Professor Kalevi Ekman	
Thesis advisors Lauri Repokari, Lic.Sc (Tech.) & Joona Kurikka, M.Sc. (Tech.)	
Date 02.05.2016 **Number of pages** 75 + (25)	**Language** English

Abstract

Designing new products and further developing existing products has become increasingly important for today's industry. Therefore, engineering education has changed from theoretical science education towards practical and challenging project-based education to teach students real-life problem-solving skills along with communication and teamwork skills needed in the present working environment of an engineering graduate. Because product development and R&D are expensive, risky, and time-consuming, industry and education around the world are interested in measuring the effectiveness of the design process and the design team. This study focuses on triangulation of three different measuring methods to understand the amount of support a design team needs from professionals in order to learn new skills for the product design process. The goal of using triangulation is to articulate the strengths and weaknesses of the three methods by comparing the collected data. Measured data is the time coaches spent with the design team during a prototyping challenge which lasted for four days and four hours and was organized at IdeaSquare in CERN in January 2016. Methods used for the data collection are time-lapse images, time-tracking software, and written coach notes. The outcome of the study is that none of the three methods proved to be superior, but each one of them brings up useful data for future studies when combined.

Keywords design thinking, measuring, product development, product design, engineering education, triangulation, ME310, paperbot, computer vision

Aalto-yliopisto
Insinööritieteiden
korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO

www.aalto.fi

Diplomityön tiivistelmä

Tekijä Jani Kalasniemi	
Työn nimi Osallistujadatan kerääminen tuotekehitysprosessissa – Kolmen erilaisen lähestymistavan vertailu	
Koulutusohjelma Konetekniikka	
Pää-/sivuaine Koneensuunnittelu	**Koodi** K3001
Työn valvoja Professori Kalevi Ekman	
Työn ohjaajat TkL Lauri Repokari & DI Joona Kurikka	
Päivämäärä 02.05.2016 **Sivumäärä** 75 + (25)	**Kieli** Englanti

Tiivistelmä

Uusien tuotteiden luominen ja olemassa olevien jatkokehittäminen ovat nykypäivän teollisuudelle yksi tärkeimmistä toiminnoista. Tästä syystä insinöörikoulutus on muuttumassa teoreettisen tieteen opetuksesta kohti käytännönläheistä ja haastavampaa projektipohjaista koulutusta. Näin pyritään kehittämään opiskelijoiden käytännön ongelmaratkaisukykyjä yhdessä kommunikoinnin ja ryhmätyötaitojen kanssa. Nämä ovat oleellisia taitoja, joita tarvitaan nykypäivän työelämässä menestymiseen. Tuotekehitys on kallista, aikaa kuluttavaa ja riskialtista, joten teollisuudessa ja koulutuksessa yhdistyy vahva tahto tuotekehityksen ja tuotekehitystiimien tehokkuuden mittaamiseen. Tämä tutkimus keskittyy vertailemaan kolmea erilaista tapaa mitata koulutettavan tuotekehitystiimin tarvitsemaa ammatillisen tuen määrää uusien tuotekehitystaitojen koulutuksessa. Vertailemalla kerättyä dataa haluttiin löytää menetelmistä eroavaisuuksia ja vertailla menetelmien vahvuuksia ja heikkouksia. Kerätty aineisto koostuu työympäristössä tehdyistä erilaisista mittauksista, joilla pyrittiin selvittämään asiantuntijoiden viettämää aikaa ja käsiteltyä aihetta kunkin kehitystiimin kanssa. Prototypointihaaste järjestettiin CERN:ssä IdeaSquarella Tammikuussa 2016. Tiedonmittausmenetelmät sisältävät automatisoitua aikajaksovalokuvausta, työajanseurantaa ja asiantuntijoiden käsin tekemiä muistiinpanoja. Tutkimuksen lopputuloksena todetaan, että yksikään näistä menetelmistä ei osoittautunut ylivoimaiseksi toisiin menetelmiin verrattaessa, vaan yhdistettäessä jokainen näistä menetelmistä tuo hyödyllistä tietoa tutkimukseen.

Avainsanat design thinking, mittaaminen, tuotekehitys, insinöörikoulutus, triangulaatio, ME310, paperirobotti, konenäkö

Acknowledgements

Getting this far has seemed to take an eternity. This thesis was my personal battle of Thermopylae. Now, I can finally say it is over. And focus all my attention on new challenges.

I would not be here without all my friends and family supporting me through this long and ponderous journey.

Thank you, mother and father, for inspiring me to be an engineer, for standing beside me, and understanding that it takes time to conquer the world and study at the same time.

Thank you, Kalevi Ekman, Lauri Repokari, Joona Kurikka, and Matti Hämäläinen, for helping and guiding me throughout this thesis.

Thank you, Peter Tapio, Eero Kivistö-Rahnasto, Veikko Immonen, Henri Lönn, and Pertti Berg for all the mental and physical support. Thanks to Katarina Gois for the language check of this thesis. Thanks to Design Factory and Wistec Oy for the warm-hearted environment where I could work on my thesis.

Special thanks to Lemmy Kilmister for being the inspiration to one awesome name for one of the participant teams!

Thank you, Lauri Repokari, for introducing this complex and chaotic world of design thinking. It changed my life. Now I'm also addicted to chaos.

Thank you, Valeria, for all the love and support.

Helsinki, May 2nd, 2016

Jani Kalasniemi

Table of Content

Appendixes

List of Figures

List of Tables

Glossary

Expression	Meaning
CERN	The European Organization for Nuclear Research, one of the world's largest and most respected scientific research centers. http://home.cern/
CSV	Comma-separated file format for saving data in plain text https://fi.wikipedia.org/wiki/CSV
DBL	Design-based learning
HSV	Cylindrical-coordinate representation of colors. HSV stand for hue, saturation and value http://en.wikipedia.org/wiki/HSL_and_HSV
ICT	Information and Communication Technology https://en.wikipedia.org/wiki/Information_and_communications_technology
IDEO	One of the leading design thinking companies in the world and also said to have created the design thinking management ideology. http://www.ideo.com/
ME310	Mechanical Engineering 310, a global interdisciplinary design innovation course led by Stanford University. http://web.stanford.edu/group/me310/me310_2016/
OpenCV	Open Source Computer Vision, a cross-platform open-source library of programming functions mainly aimed for real-time computer vision. http://opencv.org/
PBL	Project-based learning
PrBL	Problem-based learning
RGB	Additive representation of colors. RBG stands for red, green and blue. http://en.wikipedia.org/wiki/RGB_color_model

1 Introduction

This chapter describes what I've experienced and thought while doing this thesis. It introduces the background of my adventure in the design thinking world, the goal and the structure of the thesis.

1.1 Background

Design thinking and university studies have shaped me into a π-shaped person, with mechanical engineering and product design acting as the fields of deeper knowledge enhanced with a wide, though shallow, understanding on various fields such as industrial design, business, leadership, different cultures, and love.

I have always been actively interested in technology. When I was younger, I would break everything apart just to find out how it worked. Usually, this would lead to the point where I was no longer able to put it back together. When I started in elementary school, we got our first computer, and that box was fascinating. It was harder to understand how it worked since it included parts that didn't move. After years of trying to find out how to combine this insatiable hunger for understanding new technology with the ambition to create something new, I ended up in Helsinki School of Technology and finally met my conqueror.

In the second year of university, I started my studies in the vast and mesmerizing field of product development. So far, I have had the privilege to see product design from the traditional engineering point of view, as well as integrated with the ever-changing and fuzzy design thinking methodology. Product design motivated me to try new things and gather valuable experiences from different challenges. More, it inspired me to dive deeper into the challenging and intriguing world of product design development.

Back in 1950, Alan Turing was struggling with question "Can machines think?" and invented *the imitation game*, known today as the Turing test, which still today is measuring how intelligent computers are. In the same study, Alan introduced a concept of *Learning Machines* and introduced the idea of a *digital machine* known today as a computer. [1] I can just ponder how many times Turing tried, failed and tried again. If trying and failing made him one of the most impressive fathers of the modern computing, I feel I am on the right track; at least from the failing point of view.

"Anyone who has never made a mistake has never tried something new."

– Albert Einstein

1.2 Goal of the Thesis

Product design is one of the key functions for industry to design and produce new products and develop existing products. Therefore, engineering education has changed from theoretical science education towards practical and challenging project-based education. Because product development and R&D are expensive, risky, and time-consuming, industry and education around the world are interested in measuring the effectiveness of the design process and the design team. How to know that the product design team is performing its best and what kind of possibilities there are to support them on their design journey?

We will be using triangulation of three different measuring methods to understand the amount of support a design team needs from professionals when learning new skills for product design process. We will focus on teaching mechatronic skills to the students of a global innovation course ME310 through PaperBot challenge. Students will start their learning of basic electronics and coding from approximately level zero. The purpose of the PaperBot challenge is to elevate students' level of prototyping one step further.

The goal of using triangulation is to articulate the strengths and weaknesses of the three methods by comparing the collected data. I wish this study and its results will be used in engineering education and for industrial purposes in the future.

1.3 Structure of the Thesis

This thesis is divided into six sections; Introduction, Research Context, Methods, Results, Conclusions and Discussion. To help navigating through the document, each of these chapters will begin with the same description as bellow.

Introduction:

This chapter describes what I've experienced and thought while doing this thesis. It introduces the background of my adventure in the design thinking world, the goal and the structure of the thesis.

Research Context:

This chapter concludes the context in which the data gathering is executed. The chapter is a walk-through in the design thinking field and ME310, on triangulation explanation, and on PaperBot challenge concept. This chapter ends with a view on how teams are measured.

Methods:

This part explains how the data collection was executed. Where the challenge was held, who participated in this challenge and how the data was in practice gathered. Also explaining what kind of tools and parts participants had on hand.

Results:

This section of the thesis introduces the results on how much data each method generated, what kind of data and with what quality. Data is handled in three different packages picture data, time-tracking data, and coach note data.

Conclusions:

This part explains the more detailed interpretation of results and what kind of problems emerge from the data and how these problems were encountered or fixed. Data is handled in three different packages the same way as in results section; picture data, time-tracking data, and coach note data.

Discussion:

In this part, the learnings, the findings, and the personal feelings during the study are described through. The next steps of this study include a vision of an improved data gathering system and the interesting questions for the future studies that arose during this study.

2 Research Context

This chapter concludes the context in which the data gathering is executed. The chapter is a walk-through in the design thinking field and ME310, on triangulation explanation, and on PaperBot challenge concept. This chapter ends with a view on how teams are measured.

2.1 Project-, Problem- and Design-based Learning

Project-based learning (PBL) is described as learning through the complex challenge that generally results in a realistic product, event or presentation to an audience. The ambition of project-based learning is enabling the transfer of students' learning to new problems and situations together instead of only developing their content knowledge. [2]–[4]

Problem-based learning (PrBL) can be seen as a close relative for project-based learning where students work together and investigate essential problems in order to identify what they need to learn to find a solution to the problem. Problems are not fully formulated, but rather like the real world problems with multiple solutions and methods to reach a solution. [2], [4]

In *design-based learning* (DBL), students learn about the possible solution through repeating iteration cycles of *(re)defining – creating – assess*. Design-based learning can be found from various disciplines. [2], [4]

Project-, problem- and design-based learning are all student-centered pedagogies and inquiry-based forms of education, which in practice have a lot of overlap. Project-based learning tackles problem-solving by the means of design, and for simplicity, this term will be used in this work to cover all the three teaching methods mentioned. Project-based learning is used to teach 21st-century skills, such as *collaboration* and *communication*, and to cherish *deep learning*. The 21st-century skills introduced by Trilling and Fadel [5] are listed in Figure 1. Students need these skills for work life and need learning environments that enable learning experiences through real-life situations and problems. [2]

Learning and Innovation	Digital Literacy	Career and Life
• Critical Thinking and Problem Solving • Creativity and Innovation • Communications and Collaboration	• Information Literacy • Media Literacy • ICT Literacy	• Flexibility and Adaptability • Initiative and Self-Direction • Social and Cross-Cultural Interaction • Productivity and Accountability

FIGURE 1 21ST-CENTURY SKILLS BY TRILLING AND FADEL [5]

2.1.1 Engineering Education

The purpose of engineering education is to educate engineers who can design and solve open-ended real world problems [6] and provides knowledge and skills related to the professional practice of engineering [7]. Engineering education, after World War II, was based on more theoretical science education, in which engineering is taught based only on a sound foundation of science and mathematics. This resulted in engineering graduates being considerate more theoretical than practical [8]. Engineering graduates employers expressed their concern because new engineers lack the capability and preparation to define and solve open-ended problems [9], and indicated a need for engineers who are experts in their field of studies, possess excellent communication skills, work well in a team, and are lifelong learners. [6]

As in response, to better prepare graduates for engineering practices, engineering design was increased in education, since "design, above all else, defines the difference between an engineering education and a science education" [10] and is seen as "a creative, open-ended activity which typically leads to a large number of possible solution. It requires engineers to generate and synthesize ideas into workable solutions, analyze the advantages and disadvantages of a particular solution and evaluate the relative merits of alternate options or solutions" [9].

An approach to teaching design to engineering students is teaching problem-solving methods, product design processes, that students may use to confront open-ended problems [11]. Examples of this kind of methods are general product development process, modern product development process, and design thinking process. Many of these courses include hands-on learning experience to emphasize learning by doing [11]–[13]

Teaching design in engineering education can be divided four different ways: *individual-content centric*, *team-content centric*, *individual-process centric* and *team-process centric*. Individual-content centric carries many the characteristics of the so-called traditional way of teaching engineering science and mathematics [14]. An example of this kind of teaching could be a lecture where students listen to the teacher and "pouring" information to their head [15]. The team-content centric way of teaching includes and encourages collaboration and teamwork, but most of the evaluation is based on student's personal assignments and tests. These first two ways are seen as focusing more on domain specific knowledge and content. Individual-process centric way includes courses that teach the process mainly through personal homework and projects. Team-process centric approach includes courses that exploit team-based learning in teaching. [14] ME310 is an example of a team-process centric way of education.

2.2 Product Design Process

Product design process can be described as an activities or steps that need to be completed in order to transform a market opportunity into a product. Product design process is a sequence of different steps, activities, and milestones that need to be done for reaching a particular goal [16]–[18]. Some design processes are precise and detailed sequences that can be followed step by step, other processes are complex and hard to describe [6], [16], [17]. Classical and analytical design methods are applied for the development of incremental changes, whereas the design process aimed to achieve radical changes can be seen as an iteration of divergent and convergent activities [19]. This classical and analytical type of linear process can be described as a slowly closing funnel [17], whereas complicated iteration process can be described as a spiral like design cycles that iterate through prototyping, building, and testing [19].

Product design processes, such as *waterfall* development process, are also known as stage-gate development process. Distinctive characteristic, of the stage-gate process, is that after every completed stage follows a gate where the work was done is evaluated to ensure that it is worth proceeding to the next level, and the process needs to pass through all the gate to make it into production. [18]

Product design process has the same essential characteristics as product development process described by Ulrich and Eppinger [17] and modern product development process described by Otto and Wood [18], so product design process is used in this work to correspond to both of these processes.

Next three chapters clarify three different kinds of design processes and describe their characteristics.

2.2.1 General Product Development Process

Ulrich and Eppinger [17] describe in their book a general process for any product development process. It is based on six different phases, which each of them has a separate input and output and the output of each stage works as an input of the next one. These phases are called *planning*, *concept development*, *system-level design*, *detail design*, *testing and refinement* and *production ramp-up* [17].

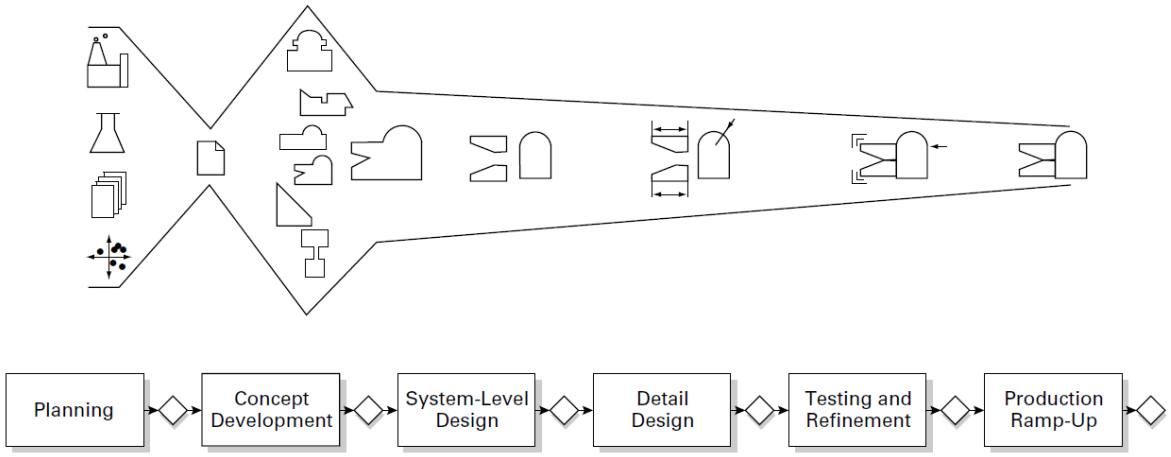


FIGURE 2 SIX PHASES OF THE GENERIC PRODUCT DEVELOPMENT PROCESS BY ULRICH AND EPPINGER [17]

Planning is the process "base zero" since it precedes the project approval and therefore launches the whole actual product development process. It begins with the identification of the opportunities guided by market situation and company's strategy and technological facilities. The output of the planning phase is a project mission statement. [17]

Concept development is a step where the need of the target market are identified, and alternative product concepts are generated. The output of this phase is one or more concepts with a description of what the product specifications, such as form, function and features of the product, might be. [17] Decisions regarding the product concept, key design parameters and possible variants of the product are to be made during planning and concept development phases [20].

System-level design phase includes a system-level definition of product's components and subsystems and early initial plans of the possible production and assembly. Outputs of system-level design phase are the first draft of the process flow for the final assembly, a geometric layout of the product and functional specification of the subsystems. [17]

In *detailed design* phase the product's geometry, tolerances, and materials are specified to the last detail together with the recognition of any standard parts used in the final product. This phase outputs a detailed documentation of the product with manufacturing drawings, tools, and machines needed for production and the definite instruction on the product assembly. [17] During the design phases the team should have made decisions regarding the components of the final product, how it will be produced and assembled, and the configuration of the physical supply chain [20].

Testing and refinement is the phase where the product is evaluated and possibly changes made by building several preproduction prototypes. Early prototypes are called *alpha prototypes*, which are built with the same product geometry and using the intended materials, but the production methods are not the ones used in the final manufacturing process. Alpha prototype's primary functions are to determine if the product design works as planned and to confirm that the product meets the needs of the key customer. Later *beta prototypes* are built from the parts manufactured by the production line, but not assembled with the final assembly process. The primary functions of the beta prototype are to test the product in the customers use environment and to test the product's performance and reliability. The output of testing and refinement phase is a production-ready final product. [17] Prototyping plan and used prototyping technologies need to be decided during testing and refinement phase [20].

The final phase of the general product development process, *production ramp-up*, the whole production and assembly of the product is made using the final production and assembly process. [17] In the final phase, decisions are made over the plan of the production ramp-up, how the final product will be tested in the market and how the product launch will be executed. [20]

General product development process is a good example of a product design process that is well documented with clear steps and phases. No surprise it is imaged as a product development funnel as seen in Figure 2. General product development process works well with solution-based problems where the primary focus of the whole process is solution driven.

2.2.2 Modern Product Development Process

Otto and Wood [18] developed an advanced product development process. Modern product development process is built from three high-level phases: *understanding the opportunity*, *develop a concept* and *implement a concept*. After these three stages, the product is ready for manufacturing. [18] All the phases and steps involved in each one of them are shown in Figure 3.

Understanding the opportunity contains all the activities needed to make the decision to launch a new product development effort. It is divided into four steps; *develop a vision*, *market opportunity analysis*, *customer need analysis* and *competitive analysis*. Developing an idea is fairly self-explaining, modern product process starts with a vision of a possible new product. Vision itself isn't really worth anything, since everyone has an idea for a new product and how something should be working. What actually makes a difference, is how a vision can be brought to life into a successful product. The second step, market opportunity analysis, is what answers if the vision would have any possibilities in today's competitive markets. After the market opportunity analysis has been done, the product development team should understand the customers', users' and stakeholders' needs regarding the product. Once team understands the needs, they should analyze and understand the competitive products on the market especially how they answer to the needs they just discovered. At this point, the design team knows the markets the product is entering, customers who are using the product and what are the technologies available for building the product. Now the design team is ready to move to the next phase, which is making the product a step more tangible and developing a new concept. [18]

Developing a concept includes all the activities to make the decision on what the product will be. This phase is divided into four steps called *portfolio planning*, *functional modeling*, *product architecture development,* and *concept engineering*. Concept development starts with portfolio planning. How the new product is positioned in the marketplace and company's portfolio of products. Second comes the step of functional modeling with specification on what the product must do to meet the needs and hopes of the customer for maximum customer satisfaction. The functional model describes the system's inputs, output, and transformations that are necessary for the product actually to work. The third step, product architecture development, in this face the series of subsets of functions are developed into a series of product subassemblies. Functional model and the possible product architecture set the base for the last step of concept development. Concept engineering is the step where the design team generates different possible ideas that implement the intended functions. At this point of the process, the vision is turned into possible product concepts. Now the product design team has a mission to choose one of the concepts that are the most promising concept, which will make it to the next phase of the modern product development process. This selected concept is still only an idea and needs to turn into a prototype by implementing the concept. [18]

Applying a concept involves all the activities to make every product work well all the time, and it is divided into four steps; *embodiment engineering*, *physical and analytical modeling*, *design for X*, and *robust design*. This is the final phase of the modern product development process, and it begins with embodiment engineering, where the concept is made tangible by specifying the components that need to be purchased or manufactured, and determine the product assembly. The second step is modeling, physically by building it and analytically by modeling the product numerically. The product is modeled and tested from various different metrics measuring the products performance in various different ways. One important step is the third phase concept implementation called 'design-for-X.' This means that the design team should think the product from several different points of views, such as design for manufacturing and assembly, design for environment and design for usability. The final step for concept implementation is a robust design. The goal is to make the product work well, make it an engineered product that is easy to assemble and manufacture, and makes sure that the performance of the product is consistent in various working environments. At this point, the design team should have a working prototype that the company needs to be evaluated and analyzed to determine whether to launch the product or to kill it. [18]

Modern product development process is still relatively straightforward, but it has already some more freedom on task execution for the design team if compared to general product development process.

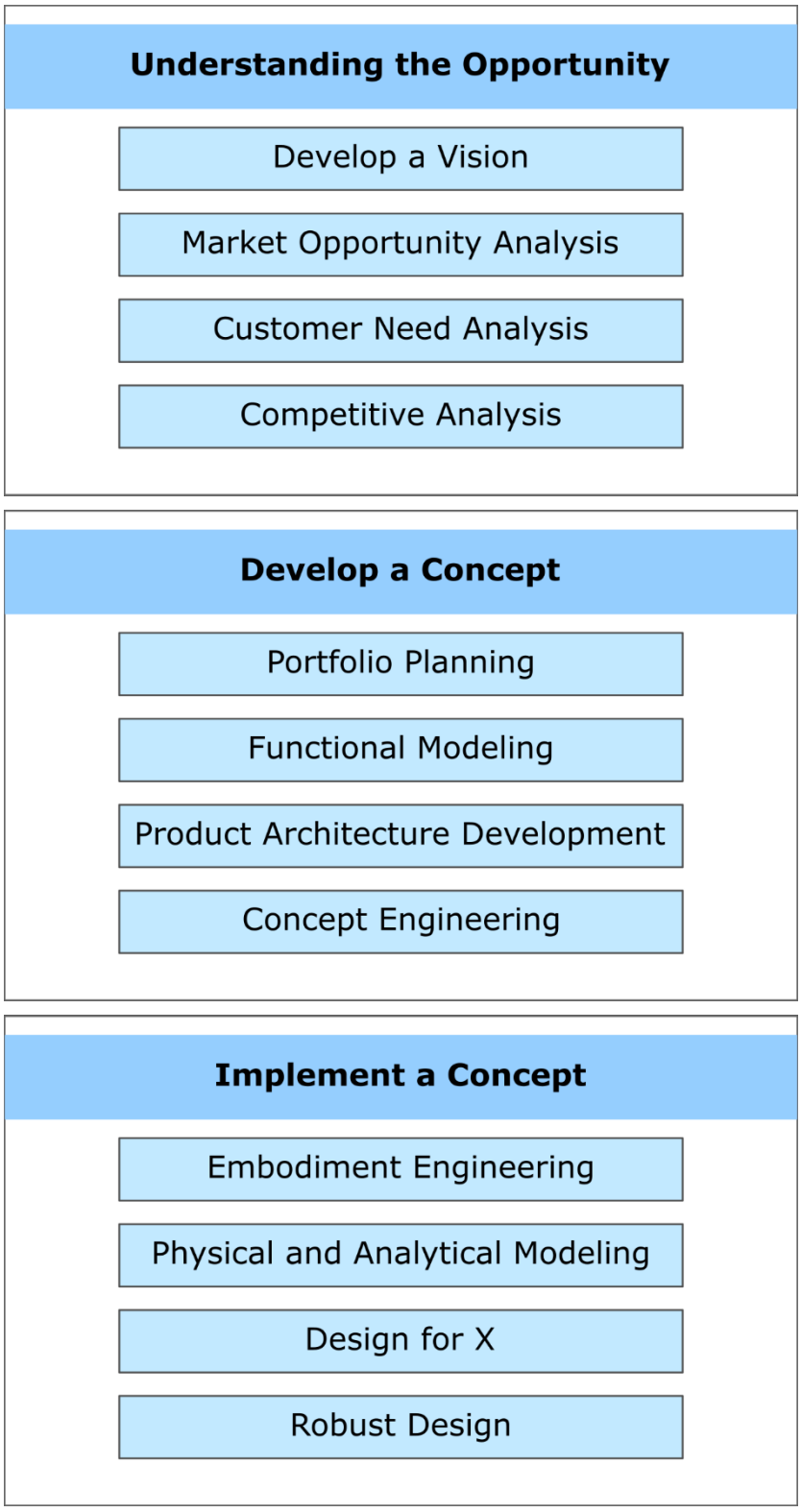


FIGURE 3 THREE PHASES WITH SUB-STEPS OF MODERN PRODUCT DEVELOPMENT PROCESS AS DESCRIBED BY OTTO AND WOOD [18]

2.2.3 Design Thinking Process

Whereas general product development process and modern product development process are examples of processes that can be described step by step, design thinking is something that is nearly impossible to describe in such a way. Author's interpretation is that the biggest difference with design thinking, to these two earlier processes, is the amount of prototyping included throughout the whole process.

Design thinking, as a term, is getting a lot of attention in the study field of design innovation. Despite to the increased amount of attention and discussion, there seems to be no one way of explaining and defining the concept of design thinking. [21]–[24] Design thinking term is used to describe and back the design's exceptional and innovative ways to solve problems. On the other hand also used to describe what designers do and how they think

Hassi and Laakso [21] searched through existing literature to find a definition for design thinking and verified the conclusion of two different streams in design thinking; design discourse and management discourse. In this thesis focus is set on the management discussion that considers the concept of design thinking as a set of practices together with cognitive approaches and a certain mindset to create innovation and value.

According to Thoring and Müller [25] "the design thinking process is determined by alternating phases of generation and selection, the environment and equipment are designed to preserve knowledge and to foster retention, the teams are able to recombine their respective expertise, and the overall culture encourages mutation of ideas and reduces the fear of making mistakes."

Tim Brown [26], who is the CEO of one of the leading design thinking companies IDEO, claims that design thinking is a discipline that uses designer's methods and sensibility to match stakeholder's needs and desires with available technology and viable business strategy together with market opportunities. Whereas product development process, introduced by Ulrich and Eppinger, answers to solution-based problems [17], design thinking is seen as a problem-based approach for solving design problems that are ill-defined, ill-structured and wicked problems [27]–[29] that have no one correct answer [30], [31].

Design thinking is emphasized as a process of continues design loops that iterate through designing, building and testing combined together with convergent and divergent activities [19]. Design thinking can feel chaotic when experiences for the first time [26] and probably this is because of the non-linear approach of design cycles. In each of these cycles, design thinking is developing a deeper understanding of the problem by trying to understand the stakeholders and users by learning from them [19]. Design thinking is described to be human-centered and user-focused design process [32]. When defining design thinking, Hassi and Laakso [32] concluded design thinking is a result of three core group of elements. These were called *practices*, *thinking styles* and *mindset*.

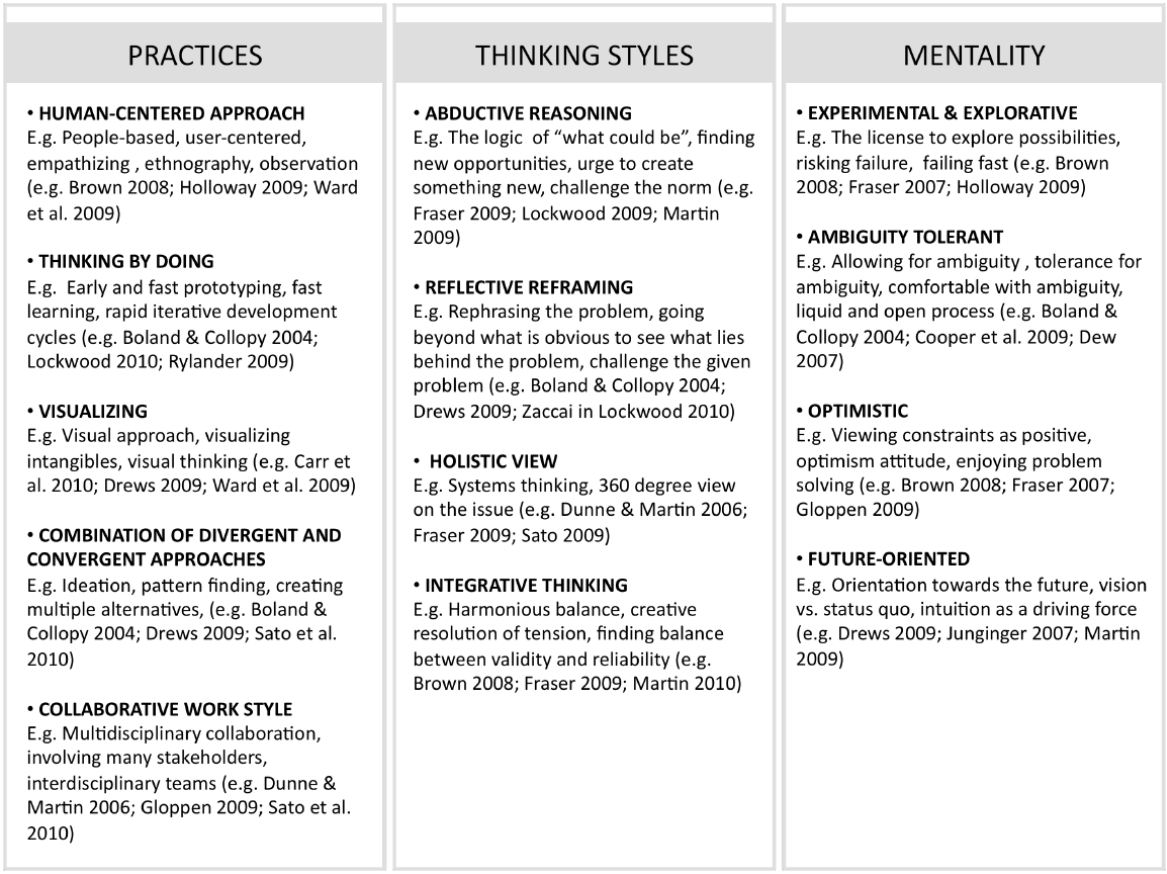


FIGURE 4 INTRODUCING COMMON ELEMENTS OF DESIGN THINKING BY HASSI AND LAAKSO [32]

Elements in a group of *practices* include *human-centered approach*, *thinking by doing*, *visualizing*, *a combination of divergent and convergent approaches* and *collaborative work style*. The human-centered approach means putting people affected by the problem first and develop empathy and understanding towards them. 'Thinking by doing' refers to the iterative and tangible personality of design thinking process that includes a vast amount of sketches and prototypes. Visualizing means expressing the problem, approaches, solutions, users, ideas and everything around the subject with tangible prototypes, pictures, drawings or anything that can be seen and touched. The combination of divergent and convergent approaches refer to widening and narrowing the view to the problem. Firstly, gather information around the subject and later find the focus to get towards the solution. [32] Divergent and convergent approach is one of the main characteristics of a design process and can constantly be found in this process in a form or another [17], [19]. Collaborative work style emphasizes that creating new big innovations is not a job of lone geniuses, but more of a work of an interdisciplinary team of design thinkers and often introduces stakeholders as a part of the design team [26], [32].

Thinking styles and mentalities that are called *abductive reasoning*, *reflective reframing*, *holistic view* and *integrative thinking*. The abductive reasoning which is logical inference starting from something that is known and exists towards something entirely new that doesn't exist and could be the answer for the problem. Reflective reframing means looking at the problem from several new point of views. Holistic view refers to a complete understanding of the problem from the environmental, cultural, user, stakeholder, customer and other people's perspective which affect to the problem. Integrative thinking means being able to combine several opposing ideas together to find a better solution than any of the separate views alone. [32]

Design thinking mindset, for both the individuals and to the organization culture, is described as being *experimental and explorative*, *ambiguity tolerant*, *optimistic* and *future-oriented*. Experimental and explorative mindset encourage taking risks by pushing the personal capacity, team's potential and technology's capabilities to the edge through trial and failure. The other mentality, ambiguity tolerance. [32] This means that anyone around design thinking has to withstand uncertainty and keep their minds open for new alternative ideas [32], [33]. Optimistic mentality strives for since the problems might feel impossible time to time, but the design thinkers need to keep a positive attitude that in the end there exists at least one solution for the problem. The future-oriented mind is expanding one's vision beyond the ordinary. [32]

Whereas Hassi and Laakso approached design thinking from a theoretical viewpoint, Thoring and Müller [25] approached from the practical point of view by observing design thinking student of ME310 in the HPI D-School in Potsdam, Germany. They propose that *practical steps*, *ideation rules and techniques*, and *mindset* are the working mechanism of design thinking.

Practical steps	Ideation rules and techniques	Mindset
• Understand and Observe • Define a Point of View • Ideation • Prototyping • Testing • Iteration	**Rules** • Be Visual • Defer Judgement • Build on the Ideas of Others • Focus on Topic • One Concersation at a Time • Engourage Wild Ideas • Go for Quantity **Techniques** • Negative Brainstorming • Dark Horse • PaperBot • Funky Prototype • Functional System Prototype	• 'I like, I wish' • 'Fail fast, fail often' • 'Think user-centric'

FIGURE 5 WORKING MECHANISM OF DESIGN THINKING BY THORING AND MÜLLER [25]

Practical steps include the six phases that are either diverging, converging, or diverging and converging. The first step is a diverging phase called *understand and observe*. This step gathers information and insights to be used as a source of material for next steps. Ideas rarely are just born out of nowhere, and a comprehensive source of material is necessary for generating ideas. The second step is *defining a point of view*, a converging phase where the information and insights from earlier phase are compressed into a problem statement to determine a possible focus for the project. Since the design thinking process is iterative, this emphasis can be changed several times during the project. *Ideation* is the third step of the process, which is a dual phase including diverging and converging. Creation of ideas concerning the problem is the diverging part, and the selection of the most promising idea is the converging part of the phase. In *prototyping* step the selected, one or more, ideas are made tangible. This phase is diverging since it will bring new knowledge of the problem and possible solution. The fifth step is t*esting*. The built prototype is tested and evaluated by users and stakeholders. Testing is again a converging phase, whereas the last step is a diverging phase called *iteration*. In this step team generates alternative solutions and improvements based on the user feedback gathered in the previous action. Iteration means also starting the process again from any of the steps explained earlier. This is why defining a straight forward process for design thinking is extremely hard. [25]

Ideation rules of design thinking process are "*be visual*", "*defer judgment*", "*build on the ideas of others*", "*focus on the topic*", "*one conversation at a time*", "*encourage wild ideas*" and "*go for quantity*". Be visual refers to remind that one picture is worth a thousand words and help in the communication of ideas and findings. Defer judgment means that everyone in design thinking should not judge any idea in any circumstance, and no one should be judged for making a mistake. Building on the top of others ideas encourages to go further and to make reinterpretations of the idea. Focus on the topic is a rule to prevent the team from losing their scope and helps the team in making choices of the ideas. One conversation at a time is to remind that every idea and opinion should be appreciated and paid attention equally. Encouraging wild ideas can offer something new that hasn't been thought before and might have a possibility to turn into a successful idea or solution. Going for quantity encourages the team to create more designs and prototypes. Ideas are allowed to be unfinished, and prototypes can be unpolished. [25]

Ideation techniques include several different methods or tasks, such as *Dark Horse*, *Negative Brainstorming*, *PaperBot*, *Funky Prototype* and *Critical Function Prototype*. Dark Horse supports the rule of encouraging wild ideas. It is a prototype of a crazy, unrealistic or even a bit dangerous idea that might have been abandoned earlier because it was not feasible or didn't fit the topic. [25], [34] Negative Brainstorming is a unique brainstorming method where only bad ideas that would make the problem even worse are to be invented and then degenerated into a positive solution [25]. PaperBot is a prototype that focuses on teaching mechatronic skills to take the team's prototyping skills one step further [34]. PaperBot is explained more carefully on page 23. Funky Prototype is a rough prototype where the whole idea of the final prototype is envisioned for the first time, and the solution for the problem begins to come reality [34], [35]. The meaning of the Critical Function Prototype is to find a small key part of the large problem and build it into a prototype. Dark Horse, PaperBot, Funky Prototype and Critical Function Prototype are four of the eight prototypes students in ME310 build during the course [34], [35]

Culture plays a vital part in design thinking mindset. This includes certain rules, like the way of giving feedback with the *'I like, I wish'* structure, the way of *thinking user-centric* and mentality of *'Fail fast, fail often.'* Providing feedback with 'I like, I wish' structure aims to increase reflection on the process and suggesting improvements to the idea. Thinking user-centric is one of the foundations of design thinking with a reminder not to develop for oneself, but focus on solving the problem for other people. 'Fail fast, fail often' mentality strives to accelerate the iterative design cycles of the process. The idea is to encourage people to do and fail instead of not doing and still failing. Thinking about mistakes as a bad thing reduces the willingness of taking risks and being afraid of taking a risk leads to something new and innovative less likely. [25]

Some of the definitions (Hassi & Laakso [32] and Thoring & Müller [25]) of design thinking are correct and complete each other. Same kind of characteristics and rules can be found, for example, on work done by Meinel and Leifer [33].

2.2.4 ME310

Mechanical Engineering 310 (ME310) is a year-long project-based capstone course for master's level students in Stanford University, Aalto University and other universities around the world. The course was originally created at Stanford University and represents a real integration of engineering, business and design disciplines. During these eight months, students learn and apply design thinking process in product development to ideate, prototype, test and iterate to with a real world wicked problems introduced by the course's corporate sponsors. [34], [36], [37]

Originally ME310 was created to provide engineering students real-life project-based challenges to meet the needs of the industry. Later the course has shifted from practical engineering experience towards the design of mechatronic systems, innovation, global collaboration and entrepreneurship. [34], [36], [37]

ME310 presents design thinking to the students as a way to approach solving complex product development problems [37]. It introduced as a simple cycle like design process of five step in a circle as seen in Figure 6. The first task is to *define the problem* which is vaguely described in design debrief created by the sponsoring company. The ME310 design process continues with *needfinding and benchmarking*. Those include activities that bring more information and knowledge about the possible problem. An important part of the design thinking is to know the possible users and stakeholders from the beginning of the process. The third step is to *brainstorm and ideate* possible solution and limitations of the solution. After ideation, the design team should pick one idea and *build a prototype* to *test* immediately and *learn* something about the problem, solution, limitation, user, stakeholder or anything related to the subject. Lastly, the process starts from the beginning again by redefining the problem, and the team continues this way until the most suitable solution have been found. Explaining ME310 design process this way might seem like a straightforward and "linear" process, but especially in the beginning, it is something totally different. These five step mix with each other and the team might jump from one step to another and skip some steps. Therefore, the right "circle" explains design thinking process better than the left one in Figure 6. However, in this type of project as the design team gains knowledge the process clarifies towards the end as the team needs to make decisions to proceed with the problem. [33], [34]

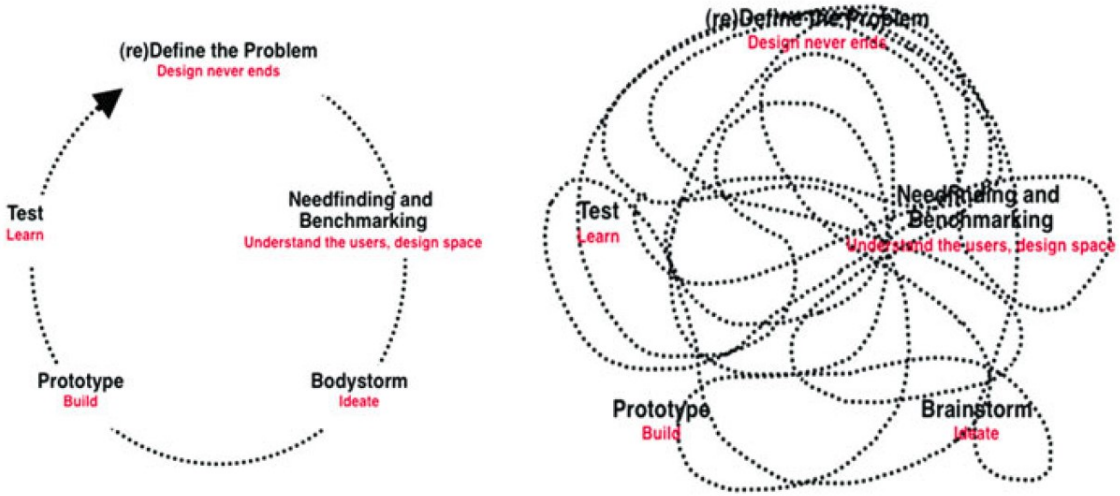


FIGURE 6 CIRCLES OF DESIGN THINKING PROCESS; LEFT THE IDEAL APPROACH, RIGHT THE REALITY [33]

ME310 can be defined as a combination of *problem-based learning*, *immersion,* and *simulation*. ME310 as a *simulator* means that ME310 is a safe environment to build prototypes, test, fail and build again. [37] "Fail fast, fail often" – the course mentality gives students more space for wildcard ideas. ME310 provides an *immersive* learning experience. Students are forced into realistic situations that really require their full attention during the time they are in the course. They need to plan and execute every detail of the project themselves, including prototyping, vendor selection, billing and such [37]. ME310 is a problem-based learning course in which students have an opportunity to work with real-life problems with industry affiliates [37].

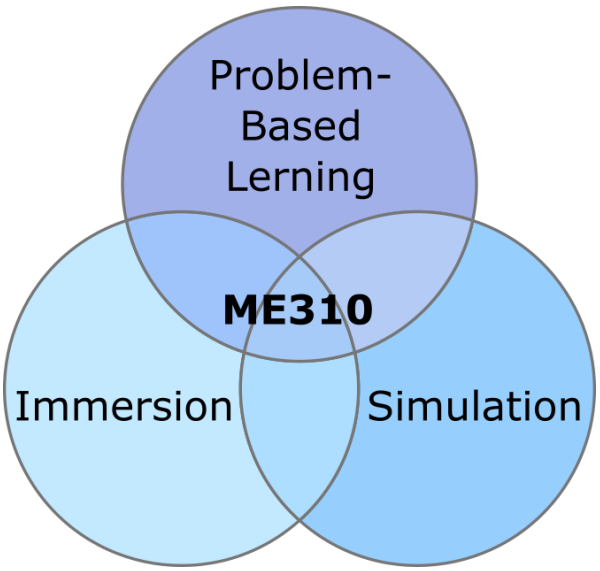


FIGURE 7 ME310 IS A DYNAMIC COMBINATION OF PROBLEM-BASED LEARNING, IMMERSION, AND SIMULATION AS DEFINED BY CARLETON AND LEIFER [37]

Stanford course's professor Larry Leifer description on ME310:

> *"ME310 is an academic year-long project-based design engineering course that began at Stanford University and has been operating continuously for over forty years. Originally created to provide engineering students with real-life engineering challenges, the course has evolved over the ages to meet the changing demands of the labor market. Over its lifetime, the course has shifted from practical engineering experience to design of mechatronic systems to design innovation and global collaboration. Meanwhile, ME310 has gone beyond the hedges of Stanford University and is now being taught in four different continents and eight different countries. The course is now focused on teaching students the innovation methods and processes required for designers, engineers, and project managers of the future. Upon the completion of the course, students have acquired the skills necessary to be global innovation leaders.*
>
> *In ME310, student teams work on innovation challenges proposed by corporate partners for eight months. Through the projects, students go through an intense and iterative process of needfinding, ideation, and rapid prototyping to create and develop new product concepts. Company involvement provides the reality that is important for teams to improve their innovation abilities. In the end, teams deliver functional proof-of-concept prototypes along with in-depth documentation that not only captures the essence of designs but the learnings that led to the ideas."* [38]

Most of the participants are completing ME310 course at their home university, and PaperBot challenge is one of the prototyping exercises that students need to accomplish during their ME310 year.

2.3 Coaching

Global product design teams, such as in ME310, work in a complex distributed environment with different time zones, cultures, and languages. This challenging environment has increased the demand for coaching to grant support to cope with the complex challenges and tasks that the design team faces during their design process. Support from the coach can range from moral support to problem solving, and there is a noticeable difference between team leading and coaching. [39], [40]

Hackman and Wageman [41] studied on coaching and team effectiveness and found out four conditions that should all be present for the coaching to augment the team effectiveness. These conditions are as follows.

1. *The group performance processes that are crucial to performance effectiveness (i.e., effort, strategy, and knowledge and skill) are relatively unconstrained by task or organizational requirements.*

2. *The team is well designed and the organizational context within which it operates supports rather than impedes teamwork.*

3. *Coaching behaviors focus on salient task performance processes rather than on members' interpersonal relationships or on processes that are not under the team's control.*

4. *Coaching interventions are made at times when the team is ready for them and able to deal with them - that is, at the beginning for effort-related (motivational) interventions, near the midpoint for strategy-related (consultative) interventions, and at the end of a task cycle for (educational) interventions that address knowledge and skill.*

Coaches are seen as a crucial part in transferring knowledge to the design team [42] and as a resource for the team [39]. Reich *et al.* [39] describe five fundamental coaching roles in ME310 as a consultant, supervisor, instructor, facilitator and mentor and a design coach need to perform various of these parts during the design process for successful coaching [40]. They also found out that different stakeholders in the design process have a different perception regarding the role of the coach and the perception changes between the different design stages [40].

2.4 Triangulation

Literally triangulation is defined as a process of using two or more known points and trigonometry to determine the location of third unknown point [43], and all the modern location systems around the world. In research, especially in social sciences, triangulation is the combination of two or more aspects of research to increase the credibility and validity of the results. Such aspects can be data sources, investigators, methodological approaches, theoretical perspectives or analytical methods. [44]

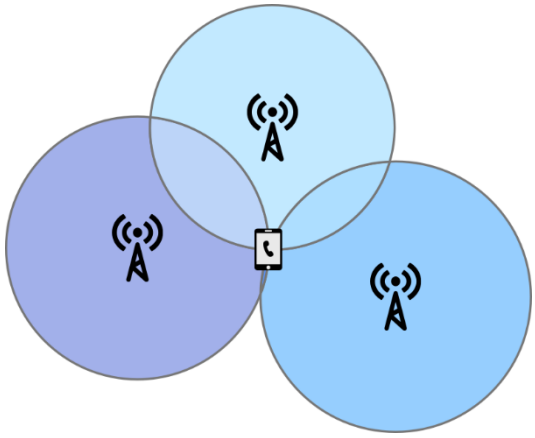


FIGURE 8 EXAMPLE OF TRIANGULATION POSITIONING (ICONS BY ROBERTO COLOMBO)

Triangulation is further divided into four types of triangulation: *Data triangulation*, *investigator triangulation*, *theoretical triangulation* and *methodological triangulation*. Data triangulation [45]. In data triangulation time, space and person can vary based on when, where and from whom the information was recorded or captured. Investigator triangulation involves using two or more observers, interviewers, coders or data analysts. Theoretical triangulation means using multiple hypothesis or theories as the base of the study. Methodological triangulation can refer to either the use of more than one data collection method or research design. Methodological triangulation can be classified further into *within-method* triangulation and *between- or across-method* triangulation. Within-method triangulation uses more than one data collection methods from the same research design approach, whereas between- or across-method triangulation use both quantitative and qualitative data collection procedures in the same research. Benefits and disadvantages of these different types of triangulation can be found in Figure 9. Based on the work of Kimchi *et al.* [46], Thurmond listed *analytical triangulation* as the fifth type of triangulation, though this sort of triangulation was not evaluated in the same way as other forms. Analytical triangulation is explained as a combination of two or more methods of analyzing data [44].

The advantages of using any kind triangulation introduced are the growing confident in the study data, creating possible new ways of understanding a phenomenon, it can reveal unique findings, integrate or challenge existing theories and provide a better understanding of the problem. On the other hand, triangulation increases the time needed to complete the research and brings up practical problems, like difficulty of dealing with the data, since triangulation gathers a vast amount of data for the research. Also, if there exists disharmony on investigators biases, conflicts of the theoretical framework or lack of understanding why and how triangulation is used, triangulation as a method can work against the study goal and debilitate the findings of the research. [44]

Data	Investigator	Theoretical	Methodological
Definition: Time, space and person vary **Benefits:** • Amount of data generated • No single information effects the study **Disadvantages:** • Difficulties trying to "fit" qualitative data into quantitative mold • What to do with singular responses or findings? • Vast amount of data	**Definition:** Two or more observers, interviewers, coders or data analysts **Benefits:** • Decrease the potential bias • Cross-checking and verifying the data will increase value of the findings **Disadvantages:** • Each investigators biases might amplify the others • Researchers might adhere to their own epistemology	**Definition:** Multiple hypothesis or theories **Benefits:** • Decrease alternative explanations • Provides boarder and deeper analysis of findings • Challenges to look beyond the obvious explanation • Can help rule out competing hypothesis • Can help prevent premature acceptance of plausable explanation • Can increase confidence in developing new concepts or constructs **Disadvantages:** • Can cause confusion if concepts within framework are • Can cause confusion if the frameworks are not identified	**Definition:** Use of more than one data collection method or research design **Benefits:** • Qualitative data may support quantitative data • May expose unique differences or information **Disadvantages:** • Differences in methods may cause conflict about research • Difficulty in meshing the numerical and • Increased expenses may be a barrier • Reluctance to publish multimethod works

FIGURE 9 DIFFERENT TYPES OF TRIANGULATIONS AND THEIR BENEFITS AND DISADVANTAGES BASED ON THE WORK OF VERONICA THURMOND [44]

In this thesis, the triangulation method is methodological triangulation since participant data has been collected with three different methods. Using multiple methods decreases the deficiencies and biases of any single method with the potential to counterbalance the flaws and weaknesses of one method with the strength of the another method [44]. By using this triangulation method, the goal is to articulate these strengths and weaknesses from each of the methods by comparing the collected data.

2.5 PaperBot

This part of the work describes the PaperBot challenge and what are the expected learning outcomes and pedagogical goals. The subject is relatively new, and there isn't much literature describing this challenge. Therefore, this chapter is supported with interviews done to people who teach and coach students in ME310[1].

PaperBot challenge is a part of ME310 course where students get a brief and funny exercise that intends to give them an introduction to the field of mechatronics. The challenge lasts for 4 days and 4 hours. PaperBot was included in ME310 course because the course projects were increasingly requiring more mechatronic skills. It is a common fact that some of the students have no experience in microcontrollers and coding, and therefore the teaching of coding starts from the basics.

For the students, this challenge is a great place to learn basic coding and find out how much it takes to actually build something with microcontrollers. Students' task is usually to create some sort of a robot, with paper and a simple microcontroller, which has to have some sort of interaction with the user. [34] The specifications for the PaperBot challenge regarding this thesis were: the robot needs to communicate with the user, the robot has to be easy to use hence don't need a user manual, the robot needs to be independently moving, and the robot has to be able to express four different emotions. Emotions are the user-centric part of the challenge, and the goal is to communicate with a real person on an emotional level. The aim is not to use the robot to manipulate the user but to create interactions with the robot that are understandable to a human. This connects the challenge strongly to one of the core components in design thinking and user-centric design in general.

[1] Personal communication, January 15th 2016
Hannula Jussi (ME310 co-instructor at Politécnico Do Porto)
Kurikka Joona (Researcher at CERN IdeaSquare & former ME310 student)
Utriainen Tuuli (Innovation Unicorn at CERN IdeaSquare & former ME310 co-instructor)
Repokari Lauri (Consulting professor & ME310 instructor at Politécnico Do Porto)

PaperBot has a special place among other challenges-ME310 students encounter. First of all, it is the first challenge after the students return from their winter holiday, and it is introduced as the challenge that will change the most the way each team prototypes. Even though every team is interdisciplinary and there are different competencies within the team, the understanding of basic electronics and mechatronics differs significantly. Therefore, the challenge begins with a lecture where the students build their first microcontroller commanded game with basic electronics components, such as LEDs, switch buttons, and resistors.

As all of the ME310 challenges have pedagogical goals, so does the PaperBot challenge. The first purpose of the challenge is to start the year again after the winter holiday. It might seem like a small thing, but in reality, this is a major deal breaker in the fact if the teams will make it through with their prototype. A good start of a new year with some new ideas and methods of prototyping is a perfect way of starting a year. The second goal is to change the way teams make and build prototypes. Before this challenge, the teams have been building low fidelity prototypes called paper prototypes. Now the fidelity of the prototypes is set higher, and mechatronic prototypes are introduced. The third goal is, once again, to bring the students out of their comfort zone. If they are not introduced to the field of microcontrollers, sensors, and electronics they will postpone implementing electronics to their final prototype for too long.

The learning experience to take from this challenge is understanding how easy it is to make small scale mechatronic prototypes with core components and microcontrollers. And, since it is not possible to have everybody interested in coding and microcontrollers, another given learning outcome is to leave a positive experience about mechatronics so that the student will understand what kind of problems could be tackled with embedded systems.

At the end of the PaperBot challenge, students did a public presentation of their work in PaperBot exhibition, and they showed how their prototype was performing.

FIGURE 10 EXAMPLES OF PROTOTYPE ROBOTS BUILT IN PAPERBOT CHALLENGE HELD AT CERN IDEASQUARE, JANUARY 2016 (PHOTOS BY PETER TAPIO)

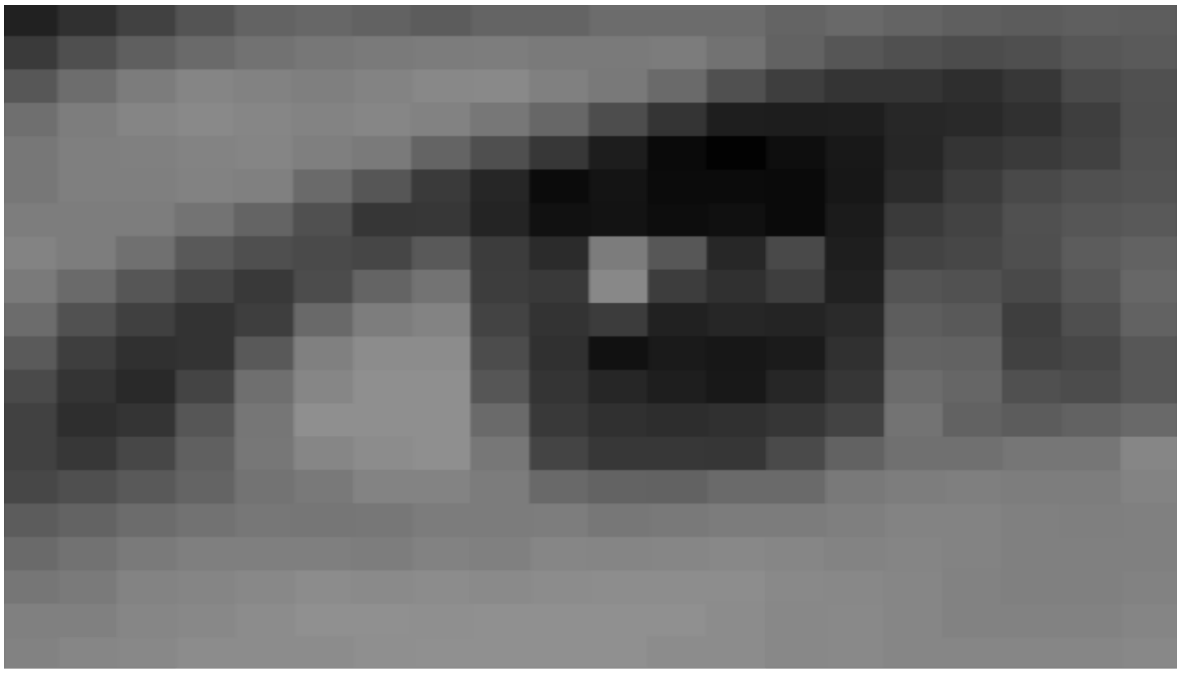
2.6 Computer Vision

Computer vision is a rapidly growing field of information technology studies with the goal to extract useful information from images or video into either a decision or a new representation. Thus, the aim is to make a computer to "see" [47]–[50]. This might sound like an easy task, but it has been proved to be much more complicated than thought [47], [49] and the dream of having a computer that could count for example all the animals in a picture seems difficult to achieve [50]. Humans and some animals understand the three-dimensional nature around us, but for a computer it is very challenging [47], [49], [50]. The visual brain performs various tricks and divides the visual data to different parts of the brain to be processed [47], [49]. This kind of preprocessing is not available when building visual algorithms [47]. Image consist of pixels where we can recognize objects and patterns, but for a computer these pixels are shown as a grid of numbers [48], [49], [51]. An example of how computers see images can be seen in Figure 11. The image is 20 x 20 pixel black and white picture of a human eye. Grayscale is easier to present with numbers than a colored image. Pixels have been enlarged to emphasize the structure of the grid.

Computer vision can be divided into three levels of processing: *low-level processing*, *middle-level processing,* and *high-level processing* [51]. Low-level processing contains primitive operations such as *reducing noise*, *enhancement of contrast* and *sharpening* of the original picture. [50], [51] This can also be known as *image processing* or *image preprocessing*, which most image analysis and computer vision algorithms require [47], [48]. Middle-level processing includes functions such as *segmentation*, *description,* and *recognition* of individual objects [51]. Middle-level processing basically means extracting relevant information from images and can also be understood mostly as *image analysis* [48], [50], [51]. High-level processes are the part of computer vision which focuses on understanding the three-dimensional world around and, in an extreme case, performing cognitive tasks that are typically correlated with human vision [51]. These are the ultimate goal of computer vision [47]–[50].

Computer vision in nowadays used in a significant number of applications, such as security, surveillance, military, medical, robotics and games [48]–[50]. Some practical examples of the use of computer vision are Google's self-driving car [52], robotic separation of waste by ZenRobotics [53] and using stereo vision for automatically detecting a spot for tree planting [54]. There are several reasons why the field of computer vision has been progressing rapidly, but the most distinct factor is the development of processing power, memory, and storage capacity [47]. One other reason is the increased use of machine learning, which today is close linked to computer vision [47], [49].

Computer vision is a vast field of different studies, but they are not explained deeply in this thesis since they are out of the scope.



33	58	86	111	118	119	126	131	122	108	90	75	65	65	70	93	107	118	127	130
48	79	109	125	128	127	123	126	106	81	63	51	47	54	79	98	112	122	129	133
64	96	122	135	127	127	127	110	86	64	48	40	51	70	91	108	124	130	133	136
83	107	133	137	130	131	114	89	70	51	51	68	87	97	97	115	126	134	137	139
99	113	131	134	133	129	99	78	57	61	89	111	117	121	115	119	126	137	141	140
104	117	126	131	127	106	80	75	76	105	127	134	142	134	122	118	127	139	144	141
99	121	132	132	123	86	53	70	100	125	141	143	144	138	130	119	126	138	144	143
92	125	136	130	100	58	55	89	115	131	140	142	144	144	132	124	129	140	144	143
100	126	135	121	79	36	37	60	61	66	75	87	105	119	125	124	127	138	144	145
101	121	129	103	55	12	17	44	57	52	48	52	58	67	105	126	134	141	144	144
107	122	122	76	28	20	19	123	136	61	16	37	49	55	98	118	133	141	144	144
109	123	106	54	9	12	13	87	60	34	26	31	43	56	98	121	134	141	144	142
108	114	81	28	2	10	16	40	48	38	23	25	48	54	103	126	137	140	140	139
100	98	63	28	16	9	9	73	62	36	26	37	55	74	106	123	134	137	137	136
105	87	50	31	23	24	26	31	32	42	47	55	67	98	122	127	131	136	137	137
100	80	52	40	37	44	59	67	84	94	100	107	116	112	126	130	133	135	133	133
96	75	45	42	51	59	66	71	82	87	100	101	97	113	128	130	131	130	131	134
91	79	55	47	58	75	80	79	73	63	65	79	95	115	126	130	128	128	130	133
95	90	74	62	65	79	86	93	87	78	71	77	96	119	124	127	127	128	131	134
93	89	80	79	80	83	88	98	103	99	87	85	106	134	132	128	129	130	133	136

FIGURE 11 EXAMPLE OF A 20 X 20-PIXEL DIGITAL IMAGE OF AN EYE AND THE GRID OF NUMBERS THAT THE COMPUTER "SEES."

2.6.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source library specialized on computer vision and machine learning. It has interfaces for C, C++, Python, Java and MATLAB and supports Linux, Max OS, and Windows operating systems. OpenCV also works on Android platforms. It was designed for real-time vision applications with the goal to serve as a simple-to-use computer vision infrastructure and contains a full general-purpose machine learning library. The alpha version of this library was released in January 1999, and presently it contains more than 2500 optimized algorithms. [49], [55]

The open source license of OpenCV is structured in such a way that anyone can use it and build commercial products without any obligations. This is one of the main reasons that OpenCV has a large user community with developers from major industrial companies and research centers, such as Google, IBM, Yahoo, Microsoft, Intel, Honda, Sony, Siemens, MIT, Stanford, and Cambridge. [49], [55] Latest estimation of a number of users is over 47 000 with over 7 million downloads throughout the world [55].

OpenCV is used in the code for image processing and analyzing because it is easy to use and offers efficient algorithms for these tasks.

2.7 Measuring Product Development Teams

Measuring team performance is a widely discussed examined topic and is studied by various different fields, such as professional sports [56], [57], military [58]–[60] and different industrial instances e.g. [61]. Measurement metrics used in this study range from metrics of the different economic point of views to physical and mental parameters of an individual and a team. The important point of measuring performance is to focus on what to measure and how to measure [62]. Hackman and Wageman [41] stated that “criterion measures in empirical research on team performance often consist of whatever quantitative indicators happen to be available or are easy to obtain (e.g., production figures for industrial workgroups or a number of correct responses for teams studied in experimental laboratories)”. Therefore, agreeing on universal design project outcome performance measurement metrics is recommended [63].

Metrics for measuring efficiency can be divided into four primary groups: *process metrics*, *program/project metrics*, *product metrics*, and *enterprise metrics*. Process parameters are short-term metrics that focus on measuring the efficiency of the product design process and can be used to predict program and product performance. Program/project metrics and product metrics are both medium-termed metrics. Program/project metrics are used for measuring the execution efficacy of the design program/project, whereas product parameters are used for measuring how the product meets the technical objectives. Enterprise parameters are long-term metrics measuring the effectiveness of the company R&D and ability to develop new products. [64]

Efficiency in product design is measured several different way. Jung and Leifer [65] introduce methods to study the relationship of design team's interactions and performance. Kress *et. al.* [66] proposes a tool to measure design team's capabilities on reframing the problem solution concept. Redelinghuys and Bahill [67] measure how resources and the effort of the design team have an effect on the creativity of the design team. Kavadias and Sommer [68] measure how organizational structures have an effect on problem-solving during the ideation phase of new product development process. Soderquist and Kostopoulos [69] study focus on the factors that affect the efficacy of new product development teams. Shah and Vargas-Hernandez [70] measure the ideation effectiveness by using the product design process and the design outcome as metrics for the study. Several studies on how to create effective teams have been comparing what key factors make an efficient design team. Reagans *et. al.* [71] study how team members' demographic characteristics (gender, race, educations, and age) and social networks have an effect on the design team. Whereas, Kress and Schar [72], [73] study team effectiveness by measuring the diversity of the design team with individual-level psychological types.

This study approaches measuring product design from the perspective of design thinking. The focus of the thesis is the product design team's learning and development by measuring the time teams needed help from the coaches during the PaperBot challenge. This study will not compare the measured data and the outcome of the challenge since the focus is on the triangulation of the methods used to collect the data.

3 Methods

This part explains how the data collection was executed. Where the challenge was held, who participated in this challenge and how the data was in practice gathered. Also explaining what kind of tools and parts participants had on hand.

3.1 Environment

According to research, space and environment has been agreed to be the biggest key factor affecting the work of a design team. Therefore, the environment should allow and support any type of ideation and prototyping activities. [19], [25], [74]

The test was done at IdeaSquare, which is test facility at CERN, the European Organization for Nuclear research. They host detector R&D projects and facilitates hackathons and MSc programs, such as Challenge Based Innovation (CBI). The layout of IdeaSquare is shown in Figure 14. The rules of IdeaSquare are as follows [75]:

1. *Be curious, be ambitious. Dream*
2. *Contribute. Collaborate*
3. *Talk to the ones you have not met before*
4. *Share your surprise of discovering the unexpected. Share your story*
5. *Cut the red tape by using scissors, cardboard, duct tape… and produce a prototype*
6. *In the workshop areas, however, cutting your fingers is not the way to cut red tape. Ask for help before you need it*
7. *Take full advantage of the Hugging Corner*
8. *Don't worry about making a mess here. The only way you create a mess is by leaving it behind unattended*
9. *Be prepared to explain what on Earth you are doing*
10. *It's always better to check the electrical wiring before you*

These rules and mentality combined with the IdeaSquare's operational aims and interests provided the perfect environment for conceptual prototyping. Hence, it was chosen as the location to organize this international multidisciplinary prototyping task.

FIGURE 12 PICTURE FROM THE PAPERBOT CHALLENGE IN IDEASQUARE AT CERN (PHOTO BY PETER TAPIO)

FIGURE 13 STUDENTS DOING GROUP WORK IN PAPERBOT CHALLENGE 2016 IN IDEASQUARE AT CERN (PHOTO BY PETER TAPIO)

IdeaSquare has two room dedicated for physical building; *Machineshop* and *Electroshop*. *Machineshop* provides tools and spaces for anything mechanical building, like cutting and drilling. Whereas *Electroshop* provides tools and spaces for electrical building, like soldering and electrical components. The kitchen is the main meeting place. Students and innovations are said to run with coffee, which makes the kitchen the heart of IdeaSquare. All the lectures were held in the main lecture area next to the red double-decker meeting room.

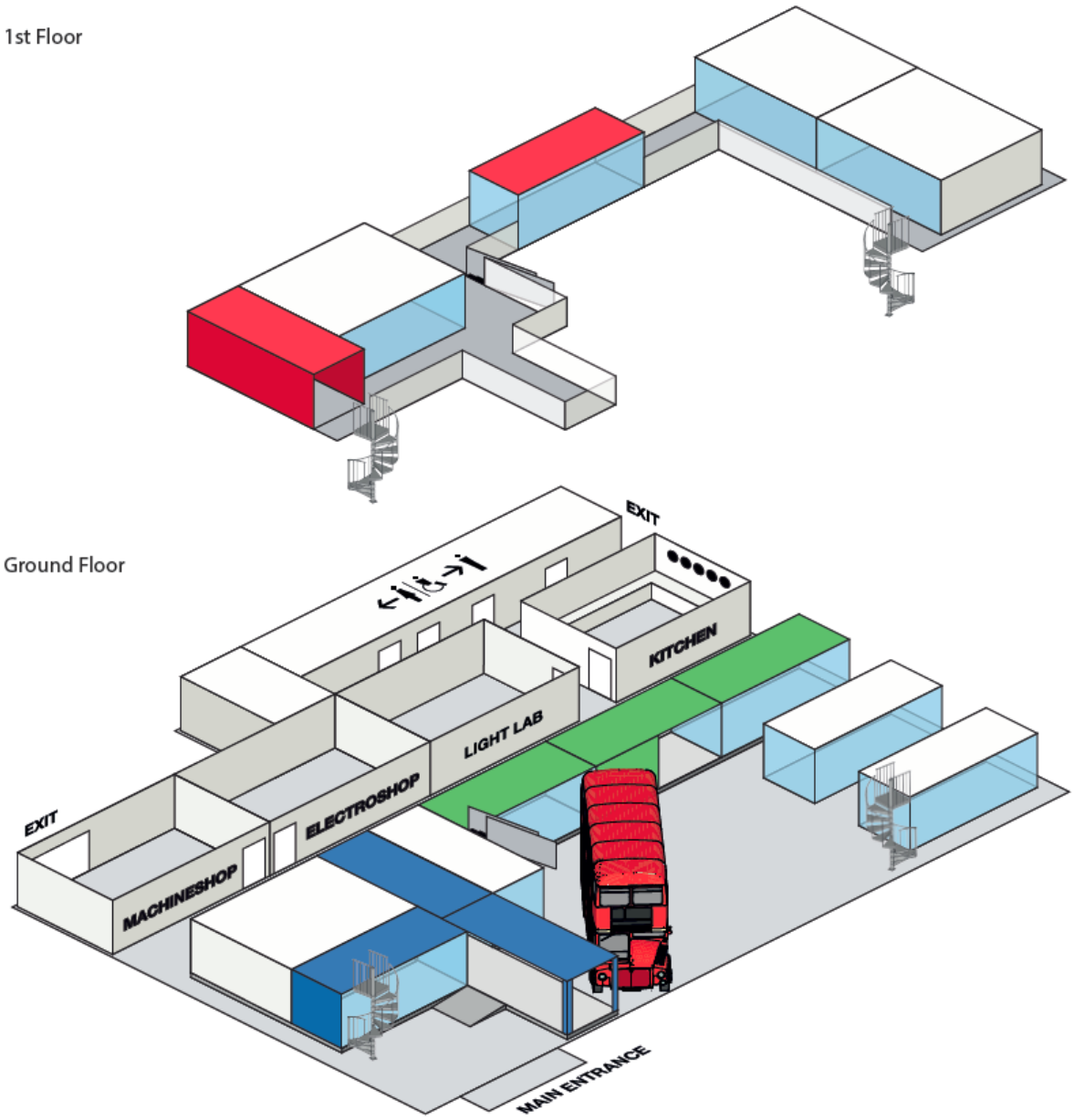


FIGURE 14 LAYOUT OF IDEASQUARE (COPYRIGHT MARIA SOLOVJEW, IDEASQUARE, 2015)

3.2 Participants

Participants were from four different counties; five from Estonia, four from Italy, four from Norway and 21 from Portugal. Participants' nationalities were Estonians, Finnish, Italians, Norwegians and Portuguese. The age difference was from 14-year-old to 45-year-old. The average age of the participants was 25 years with a median of 24 years. 14 (41.2%) participants were women, and 20 (58.2%) were men. Participants' study backgrounds were mechanical engineering, electrical engineering, automotive engineering, civil engineering, industrial engineering, industrial design, graphic design, product design, equipment design, architecture, multimedia, business, three high school students, one middle school student and one unknown. Of these subjects 24 were participating ME310 at their university.

Altogether there were 34 participants, and they were divided into nine teams. Seven of these teams were four-member teams, and two teams were three member teams. In the search for a CERN-connection, team names were inspired by heavy elements: Teams were called Actinium, Americium, Curium, Einsteinium, Fermium, Lemmium, Neptunium, Nobelium, and Thorium. Teams were placed around IdeaSquare in a way that everyone would have their own working place, and cameras can capture their every movement.

Team locations were as follows and are marked on Figure 15:

1. 1st floor, red room: Nobelium and Einsteinium
2. 1st floor, balcony: Americium
3. Ground floor, green room: Thorium
4. Ground floor, red double-decker meeting room:
    a. Lower part: Neptunium
    b. Upper part: Actinium
5. Ground floor, white room: Fermium and Lemmium
6. Ground floor, blue room: Curium

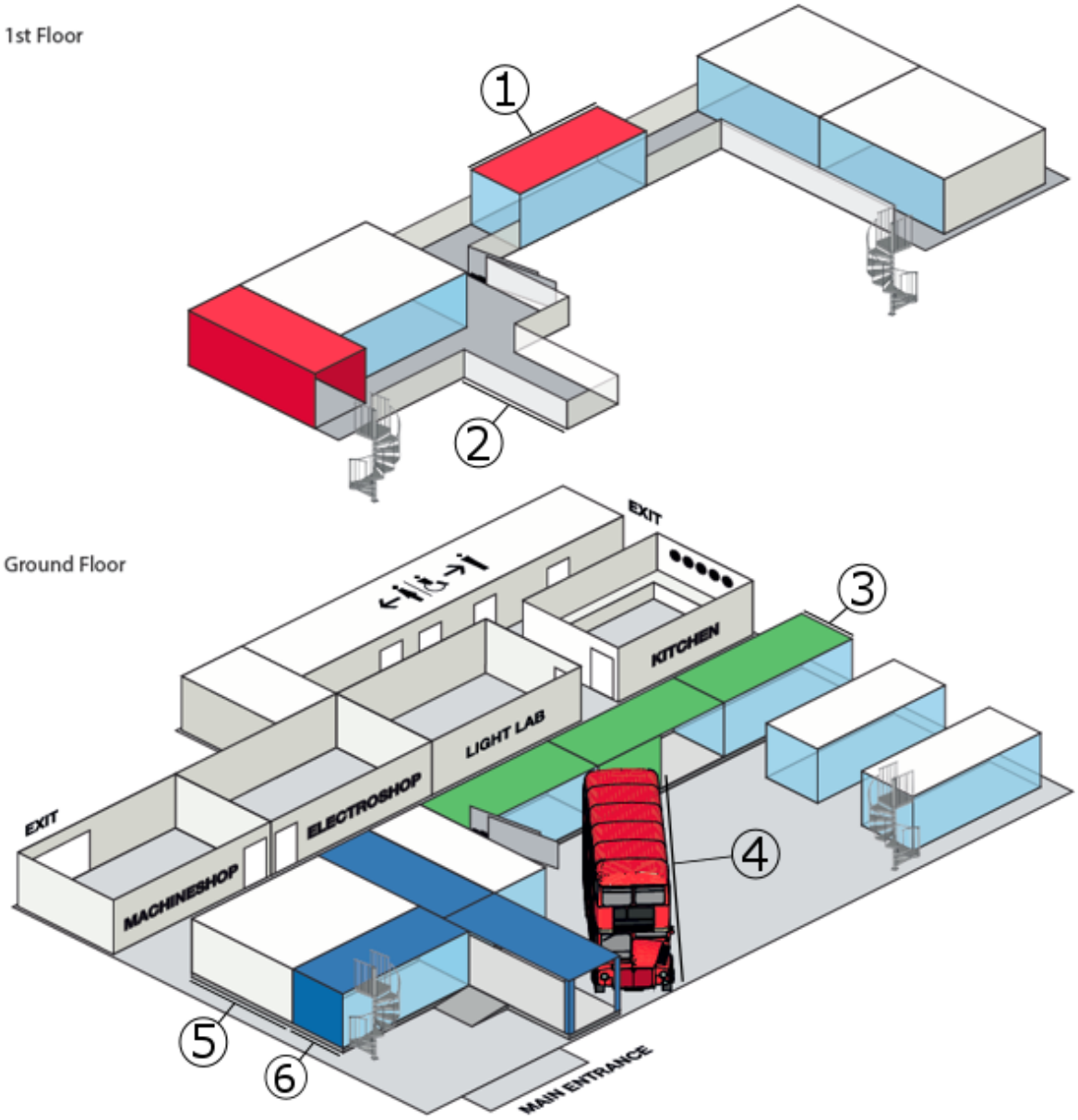


FIGURE 15 TEAM LOCATIONS AT IDEASQUARE

3.3 Equipment

Continuous time-lapse photos of teams designing, coding and building their PaperBot prototypes were taken with eight cameras. Two of those cameras were wall mounted static cameras installed to observe what is generally going on at IdeaSquare. Six of the cameras were temporarily installed action cameras. Action cameras were installed inside the teams working space and powered by either a mobile power bank or connected to a power socket when there was one available near the camera location. Cameras were set to take a picture every 10 seconds. Every day, after the participants had left the IdeaSquare, photos were copied to two external hard drives for redundancy and to prevent the memory cards to run out of space. Memory cards used in the action cameras were 32 gigabytes Kingston microSDHC Class 10 UHS-I, which had the capacity for 43 hours of time lapse recording.

Camera locations and of which team it was taking pictures:

1. 1st floor, red room; Nobelium
2. Ground floor, lower part of red double-decker meeting room; Neptunium
3. Ground floor, upper part of the red double-decker meeting room; Actinium
4. Ground floor, green room; Thorium
5. Ground floor, white room; Fermium and Lemmium
6. Ground floor, blue room; Curium
7. Wall-mounted camera (above the main lecture area); Einsteinium (also the main lecture area)
8. Wall-mounted camera (above the main entrance); Americium (also coach headquarters, aka. Arduino Bazaar)

Time data was also collected manually with mobile phones as a second method to measure the time coaches spend with the teams. Each one of the three technical coaches had cell phones with a time-tracking software installed. Two of the phones had Android operating system, and one had Sailfish operating system, which enabled using the same time-tracking software in all of the mobile phones.

3.4 Software

3.4.1 Time-tracking Software

A software called Gleeo Time Tracker was utilized in the mobile phones. Gleeo Time Tracker is a free time-tracking software for mobile phones running an Android operating system, and it is a simple tool for project and time-based time recording [76]. It allowed exporting the time data as CSV format in order to analyze the data later on. Sailfish is fully Android compatible, which made it possible to use the same time-tracking software in all the mobile phones. Each team was written as a separate task to the software settings and every time a coach helped a team he started the counter for the corresponding task. When he stopped helping the team, he stopped the tracker.

3.4.2 Computer Vision

The script goes through every picture and tries to find the color of the beanie. After finding it, the script calculates the size of the area. If it is too big or too small, it is ignored. The script cannot distinguish which coach is helping the team.

The way the code is built with OpenCV is divided into seven steps: *Load an image & read the image date and time*, *change the image from RGB to HSV color mode*, *smooth the picture*, *build a mask*, *clear noise from the mask*, *find the mask's contours*, and *write the gathered data*.

Loading an image is planned to happen automatically to all the images in the folder by just giving the file location as an argument to the script. The script will load one image at a time from this folder and after loading the image to memory, the script reads the image's modification timestamp, which is the date and time when the image was taken.

After reading the date and time, the script will change the image's color mode from RGB to HSV. HSV stands for *hue*, *saturation* and *value* and it is a cylindrical-coordinate representation for colors. This color model was used because it is easier to work with in this application.

Smoothing reduces the image's information content, in other words, blurs the image. The edges don't need to be sharp since the task is to find a color within a particular color range. Smoothing is a way of removing possible noise that will be created when building the mask.

Next, the script will go through the picture pixel by pixel and see if the pixel's color is within the specified color range. If the color is inside this range, the corresponding pixel will be marked to the mask with a white color. Otherwise, the pixel will be colored black. This way the script builds a binary, black and white, mask for the image. Since the goal is not to manipulate the original image with the mask, the code will focus only on the mask from this onwards.

The mask will most probably contain some noise which is usually seen as separate individual white pixels or black holes in the mask. Now the script will remove these pixels by using methods called dilation and erosion. Dilation is adding white pixels to the edges of the mask. This will fill small holes in the mask, and it will make the mask bigger. Erosion, on the other hand, means it will remove pixels from the mask edges. This method eliminates those individual white pixel and by consequence, it will also make the mask smaller. Usually, these methods are used together. The algorithms employed in the script are called opening and closing. Opening first uses erode to remove noise and then dilation to "grow" the mask near the original size. Closing does the same thing but uses dilatation to make the holes to fill and erode to reduce the mask near the original size. [49]

After removing the noise from the mask, the script will separate different white spots from the mask and separate them to contours. The shapes that are too small or too big are discarded. If there is a right sized silhouette, the script will save a copy of the image to a separate folder previous defined. This is just a backup procedure to check how the script performed and for debugging.

The last step is to write the gathered data into two different CSV (comma-separated values) files, one file which includes the image information from all the pictures and another which contains information from only the images that were recognized to include one or more right sized contours.

The goal of this thesis was not to generate a perfectly working script. Instead, this script was meant to be used as a tool for going through the several thousands of pictures taken during the four days of the challenge.

3.5 Coaching

The main idea with the coaching was to help teams to find the solution for their problem and not to fix the problem for them straight away. If the team was really not able to come up with the solution, the coach would correct the code and then help them to understand what he had done. Teams needed to be proactive and come to the coaches with their problem. The size of the coaching team varied daily, due to the less active coaches participating joined in late or had to leave early.

Three of the full-time coaches had a technical background and were helping subjects with their technical problems related to coding and mechanical or electrical issues.

Two of the full-time coaches were helping to find solutions for practical problems; like where could we get building material, what electronics should we order more and who should the team ask assistance on a specific matter.

There was also a group of part-time coaches that went around asking the teams how they were doing and what were their problems. This was done once a day to make sure that teams were not struggling with any issue, and that they were not shy to ask help to the coaches when needed. Figure 16 shows an example situation of a team of part-time coaches listening Nobelium Team with their progress and helping them with possible problems and questions.

Coaches' tasks during the PaperBot challenge can be divided into three broad categories: *coaching*, *education,* and *maintenance*. Coaching included helping the teams with their problems and questions. Education included arranging mini-lectures about any topic, related to the challenge, that the participants wanted more information and a lecture about using a microcontroller to build a simple game. During the challenge, the coaches held mini-lectures about topics e.g. how to use libraries, how to use functions in code, how to combine code from different projects into one, and how to use the code found on the Internet. Maintenance tasks included hands-on responsibilities, such as switching the action cameras on and off, copying the images from the action cameras' memory cards after each day, recharging the power banks used to power the action cameras and taking care of the kitchen.

Coaches were keeping notes and writing down problems teams had. This was done to get qualitative data to support the quantitative data measured with the time-tracking software.

FIGURE 16 EXAMPLE SITUATION OF COACHING WITH TEAM NOBELIUM

3.6 Coach Beanies

To distinguish coaches from subjects in the pictures, coaches were wearing a neon orange color beanie when helping the teams with their problems. The color was chosen to be something that people wouldn't usually wear, so the pictures can be diagnosed faster.

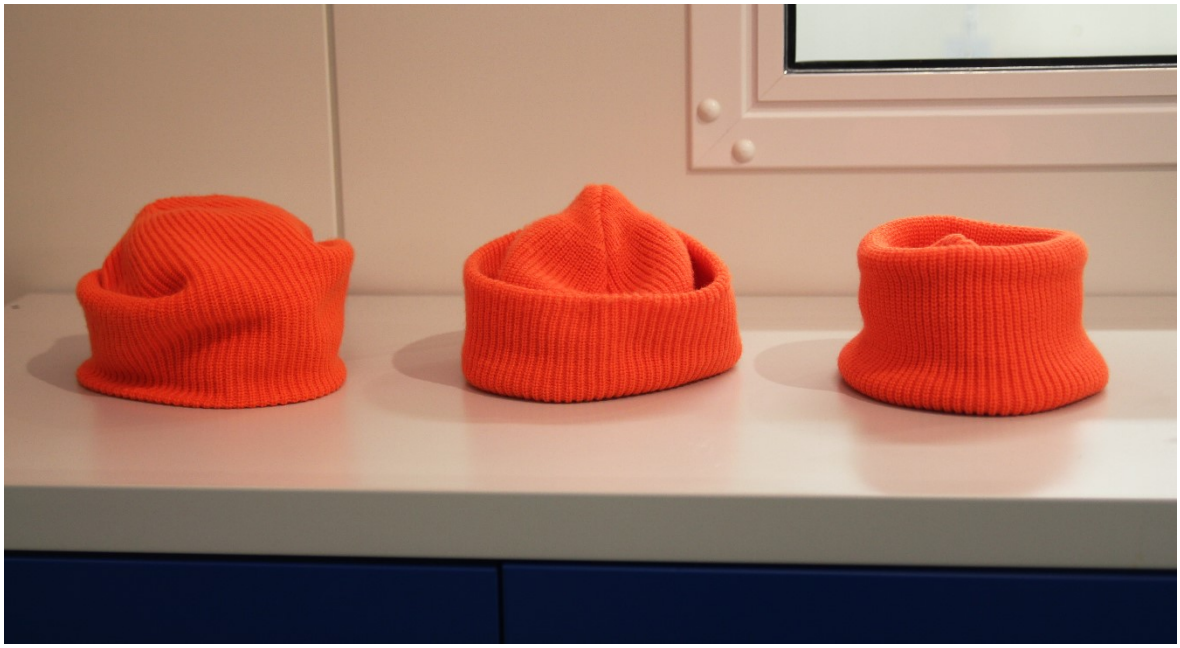


FIGURE 17 NEON ORANGE COLORED BEANIES USED BY THE COACHES (PHOTO BY PETER TAPIO)

3.7 Arduino Bazaar

Arduino Bazaar worked as the coaches' headquarters. Participants could go there and find help with their code and ask for the sensors they wanted to use. Arduino Bazaar also had two different type of 3D printers, Form 1+ produced by Formlabs and X400 PRO produced by German RepRap GmbH, that participants could use with the coaches help.

Teams were provided with access to various sensor modules, servos, motors, LED strip and basic electronic components such as wires, resistors, capacitors, and LEDs. Instead of having a fixed pre-defined set of different color LEDs, the coaches taught the participants how to use multicolor Adafruit NeoPixel Digital RGB LED strip. It had 60 RGB LEDs per meter, and each one of them was separately programmable, which makes it useful and versatile part for prototyping [77].

FIGURE 18 ADAFRUIT NEOPIXEL LED STRIP CONTROLLED WITH ARDUINO UNO, COURTESY OF ADAFRUIT [77]

Microcontroller options for the participants were Teensy 2.0, Teensy++ 2.0 [78], Arduino Nano version 3.2 [79] or Arduino UNO [80]. The benefits of these microcontrollers are their size, the ease of use and the GPIO pins. Arduino has developed its own coding environment, Arduino Software [81], which is open-sourced and multiplatform coding environment for Arduino boards. It has a good amount of prefixed functions to do the necessary commanding and basically endless list of open-source libraries to import more sophisticated functions. Teensy can use the same Arduino Software, this way participants only needed to learn one coding environment.

Table 1 shows the list of sensors that were available to the participants together with a short description about the sensor.

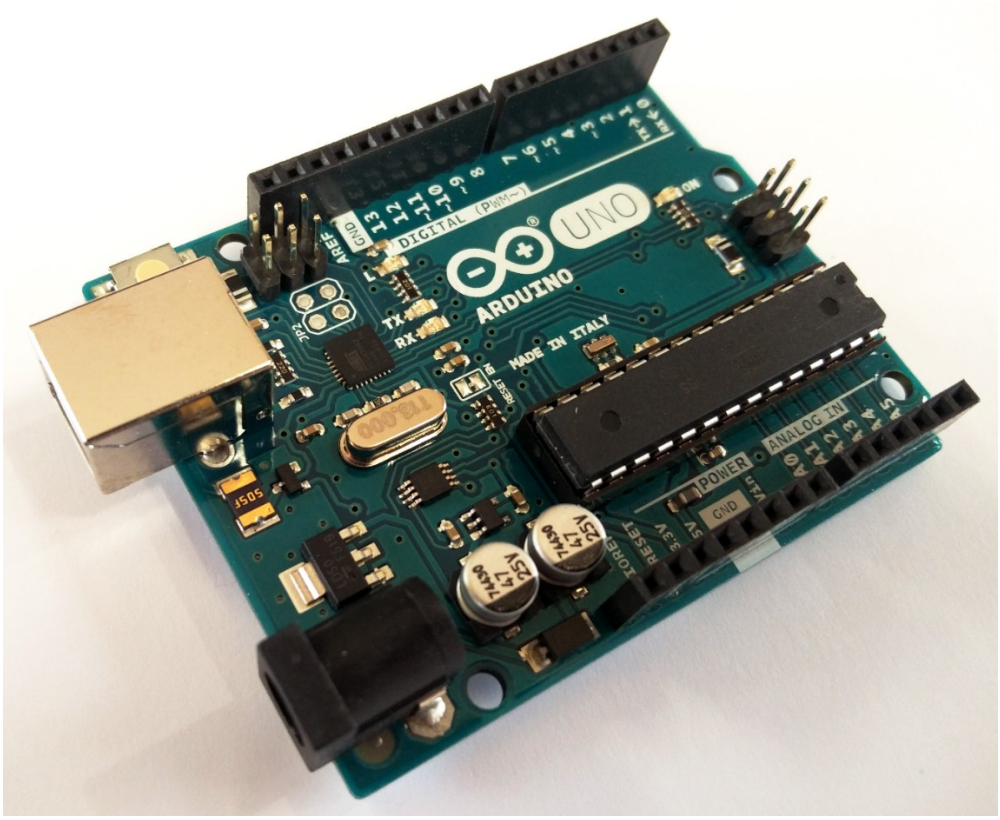


FIGURE 19 ARDUINO UNO MICROCONTROLLER

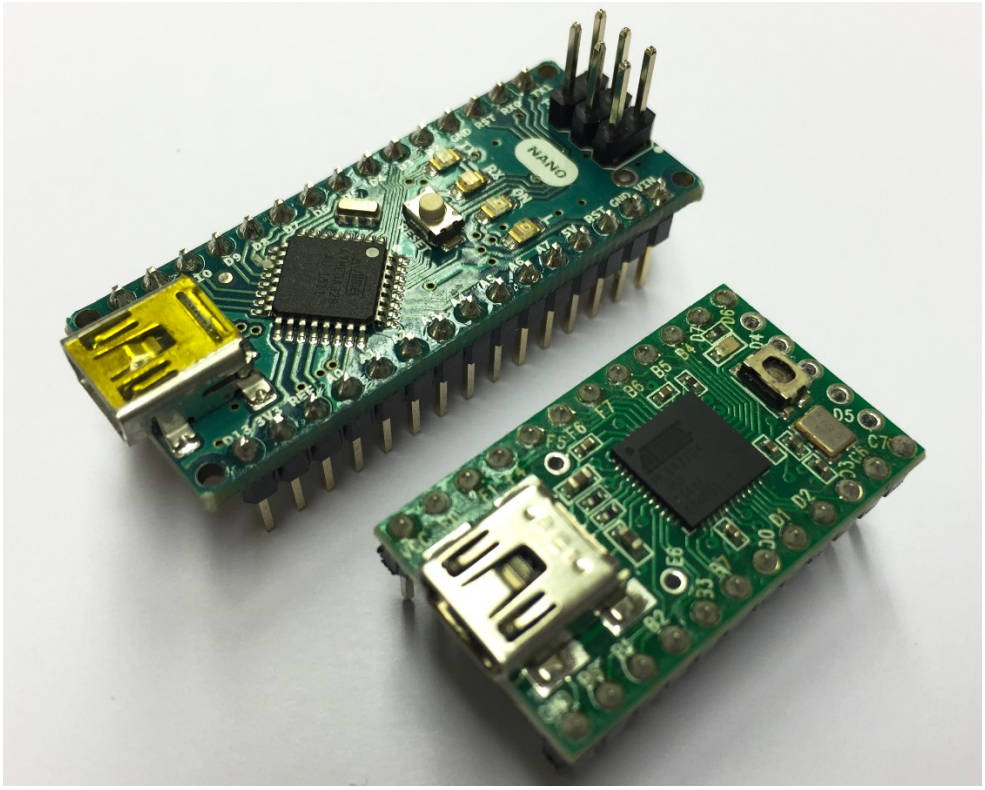


FIGURE 20 ARDUINO NANO (BACK) AND TEENSY 2.0 (FRONT) MICROCONTROLLERS

TABLE 1 LIST OF THE SENSORS AVAILABLE FOR THE PARTICIPANTS

Modules available for the participants	
Joystick	KY-023 Joystick controller for controlling robot in 2D
Relay	KY-019 Control up to 240 volt relays with 5 volts
Microphone	KY-038 & KY-037 Microphone for microcontroller
Tracking Sensor	KY-033 Tracking sensor follows a black line
Flame Sensor	KY-026 Detects infrared from a flame
Linear Hall Sensor	KY-024 Detects magnetic field strength
Capacitive Touch Sensor	KY-036 Capacitive "button"
Speaker	Speaker for microcontroller
Passive Buzzer	KY-006 Buzzer for microcontroller, can change pitch
Active Buzzer	KY-012 Buzzer for microcontroller, can make only one sound
RGB LED	KY-016 LED module with changeable color
SMD RGB	KY-009 LED module with changeable color
Two-Color Led	KY-029 & KY-011 LED module with two colors
Reed Switch	KY-025 Magnetic switch
Mini Reed Switch	KY-021 Magnetic switch
Heartbeat Sensor	KY-039 Detects heartbeat from finger
7 Color Flash	KY-034 Automatically flashing 7 color LED
Laser Emitter	KY-008 Emits laser light
Button	KY-004 Switch button
Rotary Encoders Sensor	KY-040 Encoder with rotary potentiometer
Ball Switch	KY-020 Orientation actuated switch
Photo resistor	KY-018 Photoresistor sensor, adjustable threshold
Temperature And Humidity Sensor	KY-015 & DHT11 Reads temperature and humidity
Temperature Sensor	KY-001 & KY-013 & KY-028 Sensor that reads temperature
IR Emitter	KY-005 Emits infrared light
IR Receiver	KY-022 Detects infrared light
Tap Sensor	KY-002 & KY-031 Vibration sensor
Light Blocking	KY-010 Sensor that detects if something is between the detector and source
PIR Sensor	HC-SR501 Detects moving source of infrared, for example human
Gas Sensor	MQ2 Detects H2, LPG, CH4, CO, Alcohol, Smoke or Propane
Ultrasonic Sensor	HC-SR04 Distance measurement with ultrasound
Rain Sensor	Module to detect rain
Soil Moisture	YL-69 Measures soil moisture with resistance
Obstacle Avoidance Sensor	KY-032 Avoids hitting an obstacle
Hall Magnetic Sensor	KY-003 & KY-035 Detects magnetic fields
Super-regenerative Wireless	315 MHz Frequency Wireless Receiver module + Transmitter Module
Real Time Clock	DS1302 real-time clock/calendar module

The whole idea was to give the teams various different sensors and parts to work with and not restrict their building or outcome with just a few different sensors or possible solutions. The teams did not have a pre-defined budget for the parts. However, the coaches monitored the part consumption in case of gross overconsumption.

4 Results

This section of the thesis introduces the results on how much data each method generated, what kind of data and with what quality. Data is handled in three different packages picture data, time-tracking data, and coach note data.

4.1 Picture Data

All six temporary installed action cameras and the two wall mounted cameras took a total of 178 833 photos between 11th of January 2016 and 15th of January 2016. After removing some of the pictures that were taken during the night time when there were no present subjects at IdeaSquare, the total amount of relevant photos is 129 833 during the entire active recording time of 360 hours 40 minutes and 30 seconds.

TABLE 2 THE NUMBER OF ALL THE PICTURES TAKEN

Sum of Pictures	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Camera 1 Orig.	0	8783	8416	6509	3687	27395
Camera 2	179	120	1257	1033	97	2686
Camera 3	578	3258	4259	8191	1516	17802
Camera 4	579	3261	4257	8094	1511	17702
Camera 5	631	3505	4278	6999	1560	16973
Camera 6	576	3742	4247	8199	1739	18503
Camera 7 Orig.	5760	8640	8640	8640	7206	38886
Camera 8 Orig.	5760	8640	8640	8640	7206	38886
Grand Total	**14063**	**39949**	**43994**	**56305**	**24522**	**178833**

TABLE 3 THE NUMBER OF PICTURES AFTER REMOVING THE UNNECESSARY ONES

Sum of Pictures	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Camera 1	0	4392	4208	3255	1844	13699
Camera 2	179	120	1257	1033	97	2686
Camera 3	578	3258	4259	8191	1516	17802
Camera 4	579	3261	4257	8094	1511	17702
Camera 5	631	3505	4278	6999	1560	16973
Camera 6	576	3742	4247	8199	1739	18503
Camera 7	966	4416	4686	5400	5766	21234
Camera 8	966	4416	4686	5400	5766	21234
Grand Total	**4475**	**27110**	**31878**	**46571**	**19799**	**129833**

TABLE 4 CAMERA TIME DAILY AND TOTAL (UNNECESSARY IMAGES REMOVED)

Duration	Monday	Tuesday	Wednesday	Thursday	Friday	Grand Total
Camera 1	0:00:00	12:13:26	11:43:06	9:03:28	5:07:54	38:07:54
Camera 2	0:29:40	0:19:50	3:29:20	2:52:02	0:16:00	7:26:52
Camera 3	1:36:10	9:02:58	11:49:50	22:45:16	4:12:40	49:26:54
Camera 4	1:36:20	9:03:30	11:49:30	22:29:12	4:11:44	49:10:16
Camera 5	1:45:00	9:44:02	11:52:52	19:26:20	4:19:50	47:08:04
Camera 6	1:35:52	10:23:38	11:47:54	22:46:44	4:49:46	51:23:54
Camera 7	2:40:52	12:15:52	13:00:52	14:59:52	16:00:50	58:58:18
Camera 8	2:40:52	12:15:52	13:00:52	14:59:52	16:00:50	58:58:18
Grand Total	**12:24:46**	**75:19:08**	**88:34:16**	**129:22:46**	**54:59:34**	**360:40:30**

TABLE 5 CAMERA TIME COMPARED TO GRAND TOTAL (UNNECESSARY IMAGES REMOVED)

% of Duration	Monday	Tuesday	Wednesday	Thursday	Friday	Grand Total
Camera 1	0.00 %	3.39 %	3.25 %	2.51 %	1.42 %	10.57 %
Camera 2	0.14 %	0.09 %	0.97 %	0.79 %	0.07 %	2.06 %
Camera 3	0.44 %	2.51 %	3.28 %	6.31 %	1.17 %	13.71 %
Camera 4	0.45 %	2.51 %	3.28 %	6.23 %	1.16 %	13.63 %
Camera 5	0.49 %	2.70 %	3.29 %	5.39 %	1.20 %	13.07 %
Camera 6	0.44 %	2.88 %	3.27 %	6.32 %	1.34 %	14.25 %
Camera 7	0.74 %	3.40 %	3.61 %	4.16 %	4.44 %	16.35 %
Camera 8	0.74 %	3.40 %	3.61 %	4.16 %	4.44 %	16.35 %
Grand Total	**3.44 %**	**20.88 %**	**24.56 %**	**35.87 %**	**15.25 %**	**100.00 %**

TABLE 6 CAMERA TIME COMPARED TO THE CAMERA'S TOTAL TIME (UNNECESSARY IMAGES REMOVED)

% per camera	Monday	Tuesday	Wednesday	Thursday	Friday	Grand Total
Camera 1	0.00 %	32.06 %	30.73 %	23.75 %	13.46 %	100.00 %
Camera 2	6.64 %	4.44 %	46.84 %	38.50 %	3.58 %	100.00 %
Camera 3	3.24 %	18.30 %	23.93 %	46.02 %	8.52 %	100.00 %
Camera 4	3.27 %	18.42 %	24.05 %	45.73 %	8.53 %	100.00 %
Camera 5	3.71 %	20.65 %	25.21 %	41.24 %	9.19 %	100.00 %
Camera 6	3.11 %	20.22 %	22.95 %	44.32 %	9.40 %	100.00 %
Camera 7	4.55 %	20.80 %	22.07 %	25.43 %	27.16 %	100.00 %
Camera 8	4.55 %	20.80 %	22.07 %	25.43 %	27.16 %	100.00 %
Grand Total	**3.44 %**	**20.88 %**	**24.56 %**	**35.87 %**	**15.25 %**	**100.00 %**

TABLE 7 TABLE OF HOW MANY GB OF PICTURES WHERE ON THE MEMORY CARD AFTER EACH DAY

Sum of Size GB	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Camera 1 Orig.	0.00	22.50	15.50	13.10	7.07	58.17
Camera 2	0.38	0.26	2.65	2.20	0.21	5.69
Camera 3	1.70	9.93	13.10	25.00	4.90	54.63
Camera 4	1.68	9.38	12.20	23.60	4.21	51.07
Camera 5	3.90	15.10	18.40	28.80	6.71	72.91
Camera 6	1.52	10.00	11.10	21.50	4.32	48.44
Grand Total	**9.18**	**67.17**	**72.95**	**114.20**	**27.42**	**290.91**

TABLE 8 RESOLUTION OF THE PICTURES EACH DAY

Resolutions in Megapixels	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016
Camera 1	0.0	10.0	8.0	8.0	8.0
Camera 1 Orig.	0.0	10.0	8.0	8.0	8.0
Camera 2	12.0	12.0	12.0	12.0	12.0
Camera 3	8.0	8.0	8.0	8.0	8.0
Camera 4	8.0	8.0	8.0	8.0	8.0
Camera 5	12.0	8.0	8.0	8.0	8.0
Camera 6	8.0	8.0	8.0	8.0	8.0
Camera 7	0.5	0.5	0.5	0.5	0.5
Camera 7 Orig.	0.5	0.5	0.5	0.5	0.5
Camera 8	0.5	0.5	0.5	0.5	0.5
Camera 8 Orig.	0.5	0.5	0.5	0.5	0.5

TABLE 9 AVERAGE IMAGE SIZE IN MB FOR EACH DAY

Average file size in MB	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Camera 1	0.00	2.61	1.89	2.06	1.96	1.70
Camera 1 Orig.	0.00	2.62	1.89	2.06	1.96	1.71
Camera 2	2.16	2.23	2.16	2.18	2.16	2.18
Camera 3	3.01	3.12	3.15	3.13	3.31	3.14
Camera 4	2.97	2.95	2.93	2.99	2.85	2.94
Camera 5	6.33	4.41	4.40	4.21	4.40	4.75
Camera 6	2.70	2.74	2.68	2.69	2.54	2.67
Camera 7	0.44	0.38	0.43	0.37	0.47	0.42
Camera 7 Orig.	0.41	0.41	0.41	0.41	0.41	0.41
Camera 8	0.44	0.38	0.43	0.37	0.47	0.42
Camera 8 Orig.	0.41	0.41	0.41	0.41	0.41	0.41

TABLE 10 CAMERA MALFUNCTIONS

Camera malfunctions	
	Information
Camera 1	**Monday 11th:** was not taking time lapse due to camera mounting problems **Thursday 14th:** Stopped taking pictures too early, missing about 13 hours of time lapse
Camera 2	Was not functioning as planned, took only few hours of pictures per day
Camera 3	**Tuesday 12th:** Stopped taking pictures too early, missing about 40 minutes of time lapse
Camera 4	**Tuesday 12th:** Stopped taking pictures too early, missing about 40 minutes of time lapse
Camera 5	**Thursday 14th:** Stopped taking pictures too early, missing about 3 hours of time lapse
Camera 6	No malfunctions
Camera 7	No malfunctions (pictures imported from server)
Camera 8	No malfunctions (pictures imported from server)

The initial plan was to go through the pictures automatically with a computer by using OpenCV Python library. Since this failed, 87 365 photos were inspected manually one by one. When going through the pictures manually, the focus was to find out which coach was helping which team. Also, the gather of the information about how long the teams spend time in their dedicated working place was done at the same time. This would not have been possible to accomplish with the script. The main rule, when going through the pictures manually, was to keep in mind what the script might give as a result. If a coach is helping the team but is not wearing the beanie, it will not be marked coaching.

Based on the data from pictures, coaches helped the teams a total of 20 hours 25 minutes and 59 seconds. The average time a coach helps a team was 21 minutes and 19 seconds and median 6 minutes and 50 seconds. Average of coaching per coach was 7 hours and 55 minutes and 53 seconds. Average coaching per day was 4 hours 45 minutes and 32 seconds. Resolution of the time-lapse was 10 seconds.

TABLE 11 TIME COACHED BY COACH AND TEAM (PICTURE DATA)

Coaching duration	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1		**0:55:18**	**0:39:20**	**0:24:30**	**0:15:40**	**2:14:48**
Curium		0:08:48				0:08:48
Fermium		0:46:30		0:24:30	0:15:40	1:26:40
Lemmium			0:39:20			0:39:20
Nobelium						
Thorium						
Coach 2		**1:20:36**	**1:25:02**	**3:32:36**	**4:52:42**	**11:10:56**
Curium		0:29:48	0:51:30	1:27:52	1:13:12	4:02:22
Fermium					1:14:20	1:14:20
Lemmium		0:37:28		1:36:30	2:22:50	4:36:48
Nobelium		0:13:20	0:33:32	0:28:14	0:02:20	1:17:26
Thorium						
Coach 3	**0:03:10**	**1:16:54**	**0:34:44**	**3:46:20**	**1:19:00**	**7:00:08**
Curium			0:11:20	1:20:20	0:17:00	1:48:40
Fermium	0:03:10	0:06:40	0:00:50	1:40:30	0:43:00	2:34:10
Lemmium		0:31:10	0:02:40	0:43:20	0:15:40	1:32:50
Nobelium		0:39:04	0:19:54		0:03:20	1:02:18
Thorium				0:02:10		0:02:10
Grand Total	**0:03:10**	**3:32:48**	**2:39:06**	**7:43:26**	**6:27:22**	**20:25:52**

TABLE 12 TIME COACHED BY TEAM (PICTURE DATA)

Coaching duration	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Curium		0:38:36	1:02:50	2:48:12	1:30:12	5:59:50
Fermium	0:03:10	0:53:10	0:00:50	2:05:00	2:13:00	5:15:10
Lemmium		1:08:38	0:42:00	2:19:50	2:38:30	6:48:58
Nobelium		0:52:24	0:53:26	0:28:14	0:05:40	2:19:44
Thorium				0:02:10		0:02:10
Grand Total	**0:03:10**	**3:32:48**	**2:39:06**	**7:43:26**	**6:27:22**	**20:25:52**

TABLE 13 TIME COACHED BY EACH COACH (PICTURE DATA)

Usage of coaches	Coach 1	Coach 2	Coach 3	Grand Total
Curium	0:08:48	4:02:22	1:48:40	5:59:50
Fermium	1:26:40	1:14:20	2:34:10	5:15:10
Lemmium	0:39:20	4:36:48	1:32:50	6:48:58
Nobelium		1:17:26	1:02:18	2:19:44
Thorium			0:02:10	0:02:10
Grand Total	**2:14:48**	**11:10:56**	**7:00:08**	**20:25:52**

TABLE 14 COACHING TIME COMPARED TO GRAND TOTAL (PICTURE DATA)

Time coaching %	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	0.00 %	5.76 %	7.81 %	1.72 %	3.20 %	18.49 %
Coach 2	0.00 %	5.65 %	7.21 %	16.82 %	20.50 %	50.17 %
Coach 3	0.22 %	7.11 %	2.43 %	16.03 %	5.53 %	31.33 %
Grand Total	**0.22 %**	**18.52 %**	**17.46 %**	**34.56 %**	**29.23 %**	**100.00 %**

TABLE 15 COACHING TIME COMPARED TO DAILY TOTAL (PICTURE DATA)

Time coaching %	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	0.00 %	31.17 %	42.25 %	9.28 %	17.30 %	100.00 %
Coach 2	0.00 %	11.25 %	14.37 %	33.52 %	40.86 %	100.00 %
Coach 3	0.71 %	22.71 %	7.77 %	51.16 %	17.66 %	100.00 %
Grand Total	**0.22 %**	**18.52 %**	**17.46 %**	**34.56 %**	**29.23 %**	**100.00 %**

TABLE 16 COACHING TIME COMPARED TO COACH TOTAL (PICTURE DATA)

Time coaching %	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	0.00 %	31.12 %	44.77 %	4.97 %	10.94 %	18.49 %
Coach 2	0.00 %	30.48 %	41.29 %	48.66 %	70.13 %	50.17 %
Coach 3	100.00 %	38.40 %	13.94 %	46.38 %	18.93 %	31.33 %
Grand Total	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**

4.2 Time-tracking Data

According to time tracker data, coaches helped the teams a total of 47 hours and 29 minutes. The average duration coach helping a team was 22 minutes and 26 seconds and median 12 minutes. Average of total coaching per coach was 15 hours 49 minutes and 40 seconds. Average of coaching per day was 9 hours 29 minutes and 48 seconds. Resolution of the time-tracking application was one minute and anything under one minute was not recorded.

TABLE 17 TIME COACHED BY COACH AND TEAM (TIME-TRACKING DATA)

Coaching duration	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	**0:04:00**	**3:49:00**	**2:31:00**	**2:36:00**	**6:39:00**	**15:39:00**
Actinium	0:03:00	0:40:00	1:18:00	0:45:00	6:16:00	9:02:00
Americium		1:25:00	0:31:00	0:49:00		2:45:00
Curium		0:13:00	0:02:00			0:15:00
Einsteinium		0:05:00		0:18:00	0:08:00	0:31:00
Fermium		0:53:00		0:24:00	0:15:00	1:32:00
Lemmium		0:03:00	0:40:00			0:43:00
Neptunium		0:11:00		0:07:00		0:18:00
Nobelium	0:01:00	0:10:00		0:13:00		0:24:00
Thorium		0:09:00				0:09:00
Coach 2	**0:05:00**	**2:43:00**	**3:34:00**	**5:32:00**	**8:03:00**	**19:57:00**
Actinium			0:18:00	0:18:00		0:36:00
Americium		0:05:00	0:40:00	1:33:00	0:32:00	2:50:00
Curium		0:30:00	0:50:00	1:24:00		2:44:00
Einsteinium		1:06:00	0:57:00	0:19:00	0:49:00	3:11:00
Fermium					4:17:00	4:17:00
Lemmium		0:37:00		1:07:00	2:23:00	4:07:00
Neptunium			0:16:00	0:40:00		0:56:00
Nobelium	0:05:00	0:25:00	0:33:00	0:11:00	0:02:00	1:16:00
Coach 3	**0:18:00**	**2:54:00**	**2:43:00**	**5:20:00**	**0:38:00**	**11:53:00**
Actinium		0:46:00		0:03:00		0:49:00
Americium			0:57:00			0:57:00
Curium			0:34:00	1:29:00		2:03:00
Einsteinium			0:07:00			0:07:00
Fermium	0:15:00	0:08:00		2:57:00		3:20:00
Lemmium		0:48:00	0:03:00	0:44:00	0:38:00	2:13:00
Neptunium	0:03:00	0:25:00	0:21:00			0:49:00
Nobelium		0:47:00	0:41:00	0:05:00		1:33:00
Thorium				0:02:00		0:02:00
Grand Total	**0:27:00**	**9:26:00**	**8:48:00**	**13:28:00**	**15:20:00**	**47:29:00**

TABLE 18 TIME COACHED BY TEAM (TIME TRACKER DATA)

Coaching duration	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Actinium	0:03:00	1:26:00	1:36:00	1:06:00	6:16:00	10:27:00
Americium		1:30:00	2:08:00	2:22:00	0:32:00	6:32:00
Curium		0:43:00	1:26:00	2:53:00		5:02:00
Einsteinium		1:11:00	1:04:00	0:37:00	0:57:00	3:49:00
Fermium	0:15:00	1:01:00		3:21:00	4:32:00	9:09:00
Lemmium		1:28:00	0:43:00	1:51:00	3:01:00	7:03:00
Neptunium	0:03:00	0:36:00	0:37:00	0:47:00		2:03:00
Nobelium	0:06:00	1:22:00	1:14:00	0:29:00	0:02:00	3:13:00
Thorium		0:09:00		0:02:00		0:11:00
Grand Total	**0:27:00**	**9:26:00**	**8:48:00**	**13:28:00**	**15:20:00**	**47:29:00**

TABLE 19 TIME COACHED BY EACH COACH (TIME TRACKER DATA)

Usage of coaches	Coach 1	Coach 2	Coach 3	Grand Total
Actinium	9:02:00	0:36:00	0:49:00	10:27:00
Americium	2:45:00	2:50:00	0:57:00	6:32:00
Curium	0:15:00	2:44:00	2:03:00	5:02:00
Einsteinium	0:31:00	3:11:00	0:07:00	3:49:00
Fermium	1:32:00	4:17:00	3:20:00	9:09:00
Lemmium	0:43:00	4:07:00	2:13:00	7:03:00
Neptunium	0:18:00	0:56:00	0:49:00	2:03:00
Nobelium	0:24:00	1:16:00	1:33:00	3:13:00
Thorium	0:09:00		0:02:00	0:11:00
Grand Total	**15:39:00**	**19:57:00**	**11:53:00**	**47:29:00**

TABLE 20 COACHING TIME COMPARED TO GRAND TOTAL (TIME TRACKER DATA)

Time coaching %	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	0.14 %	8.04 %	5.30 %	5.48 %	14.00 %	32.96 %
Coach 2	0.18 %	5.72 %	7.51 %	11.65 %	16.95 %	42.01 %
Coach 3	0.63 %	6.11 %	5.72 %	11.23 %	1.33 %	25.03 %
Grand Total	**0.95 %**	**19.87 %**	**18.53 %**	**28.36 %**	**32.29 %**	**100.00 %**

TABLE 21 COACHING TIME COMPARED TO DAILY TOTAL (TIME TRACKER DATA)

Time coaching %	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	14.81 %	40.46 %	28.60 %	19.31 %	43.37 %	32.96 %
Coach 2	18.52 %	28.80 %	40.53 %	41.09 %	52.50 %	42.01 %
Coach 3	66.67 %	30.74 %	30.87 %	39.60 %	4.13 %	25.03 %
Grand Total	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**

TABLE 22 COACHING TIME COMPARED TO COACH TOTAL (TIME TRACKER DATA)

Time coaching %	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	0.43 %	24.39 %	16.08 %	16.61 %	42.49 %	100.00 %
Coach 2	0.42 %	13.62 %	17.88 %	27.74 %	40.35 %	100.00 %
Coach 3	2.52 %	24.40 %	22.86 %	44.88 %	5.33 %	100.00 %
Grand Total	**0.95 %**	**19.87 %**	**18.53 %**	**28.36 %**	**32.29 %**	**100.00 %**

4.3 Coach Notes

From the total of 89 notes of the coaches, 79 helping situations were somehow technical, and the rest 10 cases were help regarding where to get materials for building the robot and giving out components from Arduino Bazaar. From those 79 technological notes 62 cases were related to coding, and the rest 17 notes were about helping the team out with mechanical problems (such as soldering or replacing a broken sensor), problems with Arduino Software (crashing, not starting or not compiling) or general questions on how the sensor in hand can be used. And those 62 coding related notes can further be categorized into two groups. 35 helping situations on how to use a sensor, how to install a library to Arduino Software, how to use a servo and how to calibrate or fine tune it, and how to use LED strip. The rest 27 notes were related to the structure of the code and coding logics, such as how to combine code, how to use functions and cleaning up the code.

TABLE 23 COACH COMMENTING ACTIVITY

	Comments	Coached	Commenting %
Coach 1	27	45	60.0 %
Coach 2	39	42	92.9 %
Coach 3	23	40	57.5 %
Grand Total	**89**	**127**	**70.1 %**

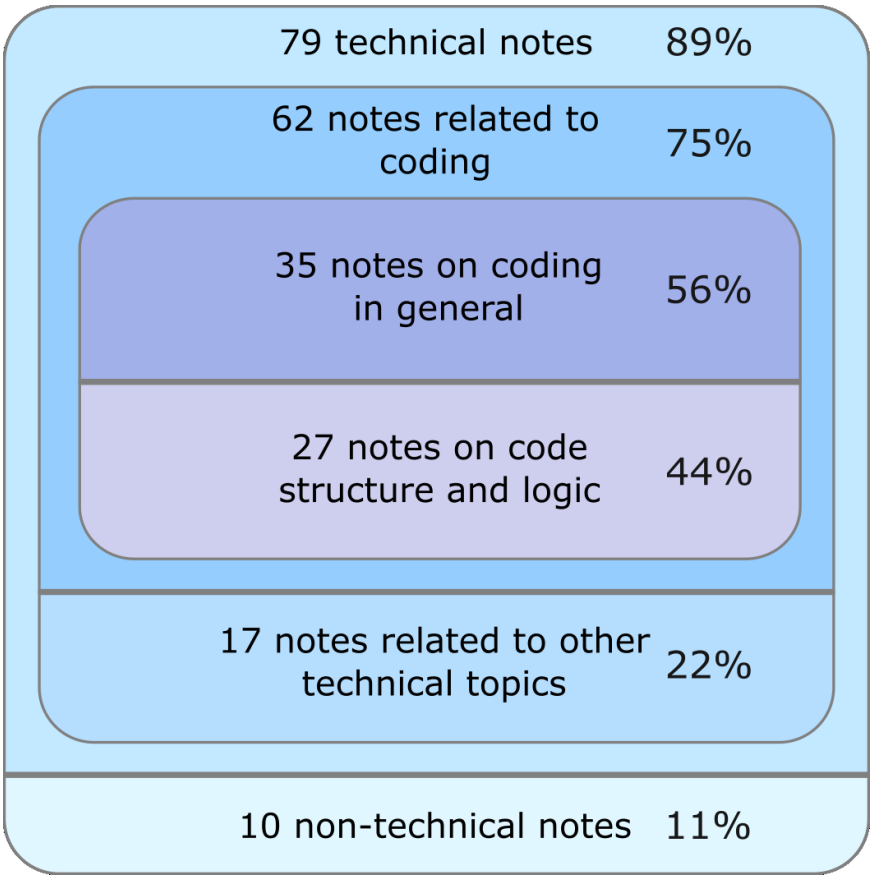


FIGURE 21 COACH NOTES DIVISION TO HIGH-LEVEL TOPICS

Deeper understanding of the records and their qualitative value are to be studied in future researches and therefore are out of the focus of this study.

5 Conclusions

This part explains the more detailed interpretation of results and what kind of problems emerge from the data and how these problems were encountered or fixed. Data is handled in three different packages the same way as in results section; picture data, time-tracking data, and coach note data.

5.1 Picture Data

A number of pictures and cameras brought up a series of different problems, and they are explained here. All code for solving these problems can be found from appendixes. There is also a code for presenting your own picture with the pixel color values as shown in Figure 11 on page 27.

5.1.1 Beanies

First and foremost, the coach beanies chosen color was not as unique as thought. IdeaSquare has a set of a different colored chair inside the rooms, and no one noticed that one of those chairs is exactly the same color as the beanies. This could have easily been corrected by removing those seats from the rooms if seen before collecting the picture data. Another issue was that the red room color was actually slightly neon. One other problem was the skin color since it seems that Eurasian skin tone has some pigments of orange. Finally, the lighting conditions in the rooms were different. This means not only that the beanie seems different colored in every room, but also that the beanie seems different colored even when moving inside the room. Therefore, the color needed to be initialized in the script separately for each room.

Another problem with the beanies was that coaches forgot to wear them all the time. Sometimes the coach would take the beanie off even though he was helping the team and other times they would even forget the beanie on the table and leave the room.

5.1.2 Cameras

Action cameras were not the same model. This resulted in a problem with a different kind of picture naming system. Most of the cameras named the pictures in the same way. They saved a maximum of 999 photos inside a folder and then created a new folder, but this made the image names almost arithmetic. There were few exceptions where the naming suddenly jumped with 10000, this happen because the time-lapse had been suspended for some reason, but the pictures still stayed in order, so it was not an actual problem. It still meant that all the pictures that were taken that one day had unique filenames. One of the cameras named the pictures with time and arithmetically rising number from 0000 to 9999. This also means that every picture taken that day had a unique number, so there was no problem there either.

One of the problems was that one of the cameras saved the same maximum 999 pictures in a folder, but when it started saving the next 999 in a new folder it would be naming the pictures from the beginning. This meant that there was more than one picture with the same filename. Analyzing the images with a script, either for the image timestamps or for searching for the beanie color, had a huge probably to run into unexpected problems.

Wall-mounted cameras and their filesystem were entirely different from the action cameras. These cameras saved the pictures to a server, and when exported to a personal computer there were almost as many folders as files itself. Some folders contained up to 999 folders which contained another 999 folders which contained a total of 8 pictures inside. Images were saved in the folder every minute, so each folder had six relevant pictures in it. Luckily the related images had a filename starting with M and a number from 1 to 6. Therefore, it was easy to gather all the relevant images inside each folder with a script.

While going through the pictures, I noticed that the pictures didn't always have a ten-second difference, even though it was set as such from the camera configuration. Time to time the difference between two of the pictures was 11 or 12 seconds. This could also be caught calculating the total amount of time from the starting and ending times of the cameras and the total number of pictures taken. The total number of photos was 129 833.There were 7 cameras taking images on one day and 8 cameras taking images on four days. Each day and camera we need to ignore on the picture and then multiply the amount with 10 seconds to find out how long the cameras were on. 129 784 pictures mean 1 297 840 seconds which corresponds to 360 hours 32 minutes and 20 seconds. When taking each camera's start and end time for each day and summing these together the time taking pictures is 360 hours 40 minutes and 30 seconds. The difference is 8 minutes and 10 seconds, which corresponds to 49 pictures. This means that the real time difference between two images was 10.00077 seconds. So the error marginal was 0.0077% between the images, and the total error was 0.00038%. This error exists because of the camera's integrated circuitry is not precise enough when calculating the ten-second difference between the pictures.

The picture data was calculated with the modification timestamps of the image file. Therefore, this error doesn't affect the data significantly. Starting timestamp was saved when the coach appeared to the picture, and the stopping timestamp was saved when the coach was not in the image anymore.

5.1.3 Camera Positioning

Although the camera positions were planned and tested beforehand in order to minimize the unwanted traffic and to have the best possible view of the team, several problems made the use of script harder than expected. Some of the rooms had big wall sized windows. Unfortunately, this also messed up the whole idea with the beanies. When coaches were moving from team to team, they would often walk by these windows, and they would easily end up in the pictures. This could have been prevented with a different camera angle or by blocking the view from the window, for example with tape and paper.

Another problem was that visitors, team members, and other people were blocking the beanie in some cases. This meant that the script couldn't see the beanie and therefore the total time of the coach helping the team would be less than it was in reality. Preventing this problem would be harder since it doesn't matter how the camera would be positioned, could always exist someone standing in front of the beanie unless there was a possibility to take pictures from above the teams. If the camera could not be placed directly above the workspace pointing down, the second best alternative would be installing the cameras as high and close to the teams as possible. Also mounting the camera to the back wall of the room would have helped. This problem would be even bigger if the coaches would have been wearing a certain colored vest instead of a beanie.

One of the cameras was installed upside down to achieve a better mounting location. This was actually not a problem, but still worth mentioning. The solution was as easy as writing a script to rotate the pictures and save them to another folder with the same filename.

One camera had the date and time set wrong, and this resulted on dating all the photos back to the year 2012. The time and date settings in every camera should have been checked to avoid this kind of a problem. In this case, it was possible to trace back the time since there was a recorded timestamp with time-tracking software when a coach was helping that team. The same camera was also positioned every day differently since there were no mounting devices for every camera. In pictures taken with this camera, (Camera 5) on Thursday 14th of January 2016, the coach was sometimes outside of the picture even though he was helping the team.

5.1.4 Camera Memory Cards

The size of the action camera memory cards was 32 gigabytes. 32 GB is the storage size represented by the manufacturing company marketing segment and uses 1000 as their base instead of 1024. This way the company actually makes 29.8 GB memory cards instead of 32 GB. Since the cameras were taking JPEG pictures, the size of the photos was in theory around 2.2 MB with 10-megapixel pictures and 1.9 MB with 8-megapixel photos. This meant that these memory cards could hold about 13 546 pieces of 10-megapixel photos or 15 685 pieces of 8-megapixel photos. This meant 37 hours 37 minutes 40 seconds with 10-megapixel photos or 43 hours 34 minutes 10 seconds with 8-megapixel photos calculated with a 10-second interval. With 5 second interval, this meant 18 hours 48 minutes 50 seconds with 10-megapixel photos or 21 hours 47 minutes 05 seconds with 8-megapixel photos. In theory, the memory cards should not get full when taking pictures for one day, but in reality, the pictures average size was bigger, as can be seen from Table 9.

Even though the memory cards were emptied every day, there were a few times when cameras run out space for pictures or for some other reason they stopped taking pictures. Also, one of the cameras was not working as it was supposed to be and for an unknown reason, it didn't manage to take pictures for longer than 3 hours and 29 minutes. The weirdest thing, with cameras running out of memory, was that one of the cameras was able to take pictures for 12 hours on Tuesday and Wednesday, but on Thursday, the camera stopped taking pictures just after 9 hours.

The memory cards used could have been bigger to prevent the cameras from running out memory, but it might be that the cameras didn't support any larger memory cards. Another way to tackle this would have been to lower the resolution of the pictures or have the whole system with cameras sending the pictures straight to the computer memory like with cameras 7 and 8.

5.1.5 Going Through the Pictures Manually

The initial plan was to go through all the photos automatically with a script using OpenCV library to find the pictures where the coaches wearing the neon orange beanie were present. Due to several problems with the images such as uneven color balance, coaches behind the window, neon orange chairs in the space, the time-lapse photos were had to be reviewed manually. Since the resolution of the time-lapse photos on cameras 7 and 8 were low quality, the focus was to go through the pictures from cameras 1-6. This meant going through 87 365 images one by one. The script planned to be used can be found in appendix 6. With some modifications to the script and to the time-lapse system, this code could be further utilized in future researches regarding this area.

This procedure took a vast amount of time and coffee for only one person. The best practice to go through the pictures with a team is dividing the images in a way that everyone has one camera to go through. An important thing to keep in mind, when splitting the workload between a team, is to make clear rules of how everything is written down. Going through over 87 365 pictures alone took me about one week of work.

The data collected was only from when a team was present at their own working place with a coach helping them. There was also some collected notes, like if there was actually someone helping, but people were in front of the camera or the coach with a beanie was actually behind the window and another kind of potential problems that might affect the output data of the script.

One question that came to mind while going through those pictures was if there was a program that would load the pictures to a computer random access memory (RAM) with a lower resolution so that it would be faster to jump from one image to another. This could possibly be done by creating a RAM disc where pictures would be first copied to there. The RAM disk could bring the reading speed of the images up to 10 GB/s when a standard hard disk drive (HDD) reading speed is somewhere between 80 – 160 MB/s. If changing from HDD to a solid state drive (SSD), the reading speed might be tripled, up to 300 MB/s. The problem was that the transfer rate of the computer USB3 port and HDD were limiting how fast one could jump from picture to picture.

5.2 Time-tracking Data

There were some sessions where the team needed help for a longer time. Time-tracking data really didn’t give any insight into how the teams needed help. That is where the written coach notes helped, to find out what was the actual problem of the team.

Time-tracking was not an exception; it had some problems as well. One of the coach's time-tracking data and note timestamps differed by one hour. The real reason that caused this issue is unclear. But our best guess is that the time from the time-tracking software has changed when the phone’s time zone changes. Data and notes were recorded and written in Switzerland, but time data was exported to Finland. Therefore, the exported time data was one hour more than the written notes.

There were different possible ways of using the time-tracking software. There were a few times when coach 3 helped several teams at the same time. Probably to be more efficient and to save him from walking back and forth all the time. He did make a remark to the software or to the notes when this occurred, but it messed up the time-tracking data. The rules should have been more carefully specified on how the time-tracking software was used, or we should have used an alternative time-tracking software that could record the time of several teams at the same time.

The precision of the time-tracking software was the biggest surprise. This time-tracking software didn't tell at any point that the minimum time for a task was one minute. This meant that every time a coach helped a team for under one minute, the software didn't mark it to the export. It might be possible to dig that information from the software's backup log. Also, the software saved only the hour and minutes when the task started and ended. Since there was a total of 127 entries, there was 254 starts and stops altogether. If all of those starts and stops were pressed precisely when a minute changes, the error of the time data would be 0%. But, since the users were humans, this was basically impossible. The maximum error is that every start and stop was pressed one second before the minute changed. This means a maximum of 14 986 (254*59) second error in total time data. The full use of the time-tracking software was 47 hours and 29 minutes, which is 170 940 seconds. Therefore, the maximum error is 8.77%. The actual error is most probably closer to 4.38%, which means an average time error of 30 seconds with every start and stop.

5.3 Coach Notes

One major problem in the notes was that there were as many ways of making notes as there were coaches. Every coach had their own way of writing things down, which made the comparison of the notes really hard. Therefore, every coach should have a template explaining how to write down their notes. In this way, the comparison of the qualitative data would be noticeably easier. The positive side is that the coaches made comments and notes with an activity of 70.1%.

5.4 Data Triangulation

When looking at the time data shown in the tables above, one can think that the method of using pictures recorded only half of the coaching situations, but this is not true. There are three main reasons why the picture data recorded not that much coaching.

Firstly, the number of photos to go through manually was enormous. And therefore, only pictures of six cameras out of eight were analyzed, the images from camera 7 and 8 will be analyzed in the future. The analyzed time-lapse covered seven teams out of nine.

Secondly, the cameras were not working as planned. Cameras' malfunctions can be seen from Table 10, where it shows that camera 2 was almost not taking time-lapse images at all, and several cameras had some sort of problems, or the cards ran out of memory (cameras 1, 3, 4, and 5). Camera 6 was the only one that functioned according to the plan the whole time, which was taking time-lapse of team Curium.

Third reason is that the teams were not always working in their dedicated working space, and therefore coaching happened around the premises. For example, team Actinium stopped using the dedicated space because the air quality wasn't the best during the working period and some of the coaching done to this team happened in Arduino Bazaar (the coach headquarter).

Since some of the picture data is missing, only the data based on teams *Curium*, *Fermium*, *Lemmium*, *Nobelium*, and *Thorium* can be compared from both data sets. Tables containing data of these five teams are listed and compared below. The data of Curium, Fermium, Lemmium, Nobelium, and Thorium are collected from corresponding tables in chapter 4 and presented here to make the comparison between the two methods easier. The percentage represents the proportional difference between the values measured from picture data and time tracker data. The percentage that was equal to 100% change or both of the values were zero, was removed from the tables.

TABLE 24 CURIUM, FERMIUM, LEMMIUM, NOBELIUM, AND THORIUM TIME DATA COLLECTED FROM TABLE 11

Coaching duration, picture data	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1		**0:55:18**	**0:39:20**	**0:24:30**	**0:15:40**	**2:14:48**
Curium		0:08:48				0:08:48
Fermium		0:46:30		0:24:30	0:15:40	1:26:40
Lemmium			0:39:20			0:39:20
Nobelium						
Thorium						
Coach 2		**1:20:36**	**1:25:02**	**3:32:36**	**4:52:42**	**11:10:56**
Curium		0:29:48	0:51:30	1:27:52	1:13:12	4:02:22
Fermium					1:14:20	1:14:20
Lemmium		0:37:28		1:36:30	2:22:50	4:36:48
Nobelium		0:13:20	0:33:32	0:28:14	0:02:20	1:17:26
Thorium						
Coach 3	**0:03:10**	**1:16:54**	**0:34:44**	**3:46:20**	**1:19:00**	**7:00:08**
Curium			0:11:20	1:20:20	0:17:00	1:48:40
Fermium	0:03:10	0:06:40	0:00:50	1:40:30	0:43:00	2:34:10
Lemmium		0:31:10	0:02:40	0:43:20	0:15:40	1:32:50
Nobelium		0:39:04	0:19:54		0:03:20	1:02:18
Thorium				0:02:10		0:02:10
Grand Total	**0:03:10**	**3:32:48**	**2:39:06**	**7:43:26**	**6:27:22**	**20:25:52**

TABLE 25 CURIUM, FERMIUM, LEMMIUM, NOBELIUM, AND THORIUM TIME DATA GATHERED FROM TABLE 17

Coaching duration, time tracker data	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1	**0:01:00**	**1:28:00**	**0:42:00**	**0:37:00**	**0:15:00**	**3:03:00**
Curium		0:13:00	0:02:00			0:15:00
Fermium		0:53:00		0:24:00	0:15:00	1:32:00
Lemmium		0:03:00	0:40:00			0:43:00
Nobelium	0:01:00	0:10:00		0:13:00		0:24:00
Thorium		0:09:00				0:09:00
Coach 2	**0:05:00**	**1:32:00**	**1:23:00**	**2:42:00**	**6:42:00**	**12:24:00**
Curium		0:30:00	0:50:00	1:24:00		2:44:00
Fermium					4:17:00	4:17:00
Lemmium		0:37:00		1:07:00	2:23:00	4:07:00
Nobelium	0:05:00	0:25:00	0:33:00	0:11:00	0:02:00	1:16:00
Thorium						
Coach 3	**0:15:00**	**1:43:00**	**1:18:00**	**5:17:00**	**0:38:00**	**9:11:00**
Curium			0:34:00	1:29:00		2:03:00
Fermium	0:15:00	0:08:00		2:57:00		3:20:00
Lemmium		0:48:00	0:03:00	0:44:00	0:38:00	2:13:00
Nobelium		0:47:00	0:41:00	0:05:00		1:33:00
Thorium				0:02:00		0:02:00
Grand Total	**0:21:00**	**4:43:00**	**3:23:00**	**8:36:00**	**7:35:00**	**24:38:00**

TABLE 26 COMPARING THE PICTURE AND TIME TRACKER DATA BY COACH AND TEAM

Data % differences	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Coach 1		**37.2 %**	**6.3 %**	**33.8 %**	**4.3 %**	**26.3 %**
Curium		32.3 %				41.3 %
Fermium		12.3 %		2.0 %	4.3 %	5.8 %
Lemmium			1.7 %			8.5 %
Nobelium						
Thorium						
Coach 2		**12.4 %**	**2.4 %**	**23.8 %**	**27.2 %**	**9.8 %**
Curium		0.7 %	2.9 %	4.4 %		32.3 %
Fermium					71.1 %	71.1 %
Lemmium		1.2 %		30.6 %	0.1 %	10.8 %
Nobelium		46.7 %	1.6 %	61.0 %	14.3 %	1.9 %
Thorium						
Coach 3	**78.9 %**	**25.3 %**	**55.5 %**	**28.6 %**	**51.9 %**	**23.8 %**
Curium			66.7 %	9.7 %		11.7 %
Fermium	78.9 %	16.7 %		43.2 %		22.9 %
Lemmium		35.1 %	11.1 %	1.5 %	58.8 %	30.2 %
Nobelium		16.9 %	51.5 %			33.0 %
Thorium				7.7 %		7.7 %
Grand Total	**84.9 %**	**24.8 %**	**21.6 %**	**10.2 %**	**14.9 %**	**17.1 %**

TABLE 27 CURIUM, FERMIUM, LEMMIUM, NOBELIUM, AND THORIUM DATA COLLECTED FROM TABLE 12

Coaching duration, picture data	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Curium		0:38:36	1:02:50	2:48:12	1:30:12	5:59:50
Fermium	0:03:10	0:53:10	0:00:50	2:05:00	2:13:00	5:15:10
Lemmium		1:08:38	0:42:00	2:19:50	2:38:30	6:48:58
Nobelium		0:52:24	0:53:26	0:28:14	0:05:40	2:19:44
Thorium				0:02:10		0:02:10
Grand Total	**0:03:10**	**3:32:48**	**2:39:06**	**7:43:26**	**6:27:22**	**20:25:52**

TABLE 28 CURIUM, FERMIUM, LEMMIUM, NOBELIUM, AND THORIUM DATA GATHERED FROM TABLE 18

Coaching duration, time tracker data	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Curium		0:43:00	1:26:00	2:53:00		5:02:00
Fermium	0:15:00	1:01:00		3:21:00	4:32:00	9:09:00
Lemmium		1:28:00	0:43:00	1:51:00	3:01:00	7:03:00
Nobelium	0:06:00	1:22:00	1:14:00	0:29:00	0:02:00	3:13:00
Thorium		0:09:00		0:02:00		0:11:00
Grand Total	**0:21:00**	**4:43:00**	**3:23:00**	**8:36:00**	**7:35:00**	**24:38:00**

TABLE 29 COMPARING THE TIME COACHED BY TEAMS

Data % difference	11.01.2016	12.01.2016	13.01.2016	14.01.2016	15.01.2016	Grand Total
Curium		10.2 %	26.9 %	2.8 %		16.1 %
Fermium	78.9 %	12.8 %		37.8 %	51.1 %	42.6 %
Lemmium		22.0 %	2.3 %	20.6 %	12.4 %	3.3 %
Nobelium		36.1 %	27.8 %	2.6 %	64.7 %	27.6 %
Thorium				7.7 %		80.3 %
Grand Total	**84.9 %**	**24.8 %**	**21.6 %**	**10.2 %**	**14.9 %**	**17.1 %**

TABLE 30 CURIUM, FERMIUM, LEMMIUM, NOBELIUM, AND THORIUM DATA GATHERED FROM TABLE 13

Usage of coaches, picture data	Coach 1	Coach 2	Coach 3	Grand Total
Curium	0:08:48	4:02:22	1:48:40	5:59:50
Fermium	1:26:40	1:14:20	2:34:10	5:15:10
Lemmium	0:39:20	4:36:48	1:32:50	6:48:58
Nobelium		1:17:26	1:02:18	2:19:44
Thorium			0:02:10	0:02:10
Grand Total	**2:14:48**	**11:10:56**	**7:00:08**	**20:25:52**

TABLE 31 CURIUM, FERMIUM, LEMMIUM, NOBELIUM, AND THORIUM DATA COLLECTED FROM TABLE 19

Usage of coaches, time tracker data	Coach 1	Coach 2	Coach 3	Grand Total
Curium	0:15:00	2:44:00	2:03:00	5:02:00
Fermium	1:32:00	4:17:00	3:20:00	9:09:00
Lemmium	0:43:00	4:07:00	2:13:00	7:03:00
Nobelium	0:24:00	1:16:00	1:33:00	3:13:00
Thorium	0:09:00		0:02:00	0:11:00
Grand Total	**3:03:00**	**12:24:00**	**9:11:00**	**24:38:00**

TABLE 32 COMPARING THE TIME COACHED BY EACH COACH

Data % differences	Coach 1	Coach 2	Coach 3	Grand Total
Curium	41.3 %	32.3 %	11.7 %	16.1 %
Fermium	5.8 %	71.1 %	22.9 %	42.6 %
Lemmium	8.5 %	10.8 %	30.2 %	3.3 %
Nobelium		1.9 %	33.0 %	27.6 %
Thorium			7.7 %	80.3 %
Grand Total	**26.3 %**	**9.8 %**	**23.8 %**	**17.1 %**

5.5 Study Outcome

Before the comparison between this five teams was done, it looked like the picture method wasn't even close to the same time as the time tracker data. But after removing the team's whose data collection was corrupted, the difference between the two methods is 17.1%. This difference can be explained with the third reason listed above; the coaches helped the teams away from their working space in areas that are not included in the pictures.

Strengths and weaknesses of the different data collection methods are shown in Figure 22.

Picture Data Method

Strengths	Weaknesses
• Time-lapse was taking pictures automatically all the time, no human error • Good quantitative data • 10-second resolution • Small error due to good resolution • Pictures can give a bit more information on why and how the coach was helping	• Images include coaching taking part only where the cameras are • Need of coding to get time data from the pictures • Other people may block the view by staying in front of the camera • Memory cards can hold only a certain number of pictures

Time Tracking Data Method

Strengths	Weaknesses
• Data collection not tied to one place • Good and reliable quantitative data • Easy export to excel with timestamps	• Human error factor, forget to start/stop • One-minute resolution • No insight on why and how the coach was helping

Note Data Method

Strengths	Weaknesses
• Qualitative data about the participants doubts and questions	• Coaches forgot to write the information down • Not all of the notes had timestamp • Resolution of the timestamps closer to 5 minutes

FIGURE 22 METHODS' STRENGTHS AND WEAKNESSES

Picture and time-tracking methods are close to as reliable in measuring quantitative data, but they clearly are missing the information on what participants were asking and what kind of problems they had during the challenge. On the other hand, note collecting method gives a good insight on the problems but has a major reliability problem with timestamps. By using note collecting together with either the picture or time-tracking method provides precise time information with good qualitative data. Still, the resolution of time-tracking method was only one minute in this study, and therefore picture method offers six times better resolution when measuring time. Picture method can measure data only where the cameras are installed. Comparing shows that none of the tested methods are entirely reliable alone, but if all of the methods are combined, the quality of the data is far more reliable.

The outcome of the study is that none of the three methods are superior, but each one of them brings up useful data for future studies when combined. As a conclusion to this study can be said that none of these three methods would provide quality data alone.

6 Discussion

In this part, the learnings, the findings, and the personal feelings during the study are described through. The next steps of this study include a vision of an improved data gathering system and the interesting questions for the future studies that arose during ~~this~~ study.

Data was gathered in three different ways, and the goal was to find out their strengths and weaknesses. It is clear that each one of them tell a different story about how much the coaches were helping the teams, and what kind of problems the teams were having. Collected data shows that the use of coach help was increasing daily towards the deadline. Also, our perceptions about understanding the roles of the coaches were clear for the team. If this kind of methods will be implemented in the corporate world in the future, clear roles for coaches will be necessary.

Pictures did give accurate information about when the team was in their dedicated working area, but this data had to be gathered manually by going through the pictures one by one. If there would be a satisfactory way of tagging the participants with beanies, vests or RFID tags, this data collection could be done automatically with a computer. This data could not have been collected with time-tracking software because that would mean a commitment from the participants by pressing start every time they arrive and stop as they leave their working place. This method would probably end up messing the data since the participants would probably forget to start or stop the timer. Therefore, this data need to be somehow collected automatically.

Pictures were also a way to find out how much coaches spend time with the teams. It was an accurate and fairly good way to collect this kind of data when going through the pictures manually. The fact is that with people staying in front of the camera, other orange colored interior elements or even furniture in the spaces messed up the possibility to gather data automatically with computer vision.

Time-tracking data was time data about how much time the coaches were spending with the teams. There were only a few times when a coach made a mistake by forgetting to start or stop the timer. It is believable that the fact only three coaches were using this time-tracking and each of them had their personal mobile phone are the reasons why there were no more forgetting. Alone this information would be quite hard to analyze since one could only see the duration of the session and time when it happened.

Notes, on the other hand, did give an enormous amount of additional qualitative information about what kind of problems the teams were facing in this challenge. The problem was that there was not an established way to write these notes. Finding a pattern from well-structured and well-written notes that would follow a template would be easier. From the notes written by coaches, it can be inferred that the participants were mostly at a suitable skill level for this type of exercise as the meaning of PaperBot challenge is to teach basic skills on coding with microcontrollers.

6.1 Next Steps

6.1.1 Future Development

If this kind of a study needed to be done in the future, few points would make the analyzing of the collected data faster and easier.

First of all, the picture system should be a central cloud-based system where each camera would be connected to a personal computer, like Rasberry Pi, BeagleBone or similar. In this way, each picture could be automatically named with a unique chosen name. Since the pictures would be straight sent to the cloud, like Dropbox, Google Drive, One Drive, own server or similar, there would not be a problem with memory cards running out of space. The computer with a camera could perform preprocessing of the pictures before saving them to the cloud, and there should be at least one camera for each group of participants. Pictures with two teams get messy when eight persons are moving around the camera.

Centered such a way that all the cameras are controlled from one master computer. It also takes away the human factor where someone would forget to start a camera or save the pictures after each day.

Even though the system would be much more reliable and more stable, it does not mean that this system could not be moved. Every camera would be a separate system transferring pictures to a specific folder wirelessly through WiFi.

One bonus feature that could be included in this kind of system would be a camera with movement detection. It would compare the camera input and only save the pictures when there is enough change between the pictures. This would greatly reduce the total amount of pictures taken and processed. This kind of system could be built by comparing the differences between the previous picture and the new picture.

Secondly, the time-tracking software precision should be at least 10 seconds instead of one minute. As calculated earlier, the assumed 30-second error meant an error margin of 4.38%. If the precision were 10 seconds instead of one minute, the error margin would be a maximum of 1.34% and in reality closer to 0.67%.

Thirdly, the template for the notes as told. It would make it easier and a lot faster for the coaches to write their data down. In this way the activity percentage could be even bigger than the 70.1% we had this time. Written notes could also be replaced with recorded voice notes marked with timestamps.

Additionally, more advanced computer vision techniques such as machine learning, facial recognition, and image classification can be used to gain additional information. Potentially these systems could be used to recognize the movement of individuals within the target space and possibly even classify certain activities.

Lastly, the main rules for handling the data. Always have a separate backup of all the raw data and never mess up with this information before really needed. Do not change the original data and files, but make a copy of it if there is a need to change something, like rotate the picture. Build a system that saves the images from different cameras to different folders with an identification number, date, time and arithmetically changing index number and save each day data to its own folder. In this way, the system is ready for the different type of automated data collection with a computer.

More ideas and possible ways of improvement could be studied from design observatories, which are widely used around the world for collecting data from designers and designing situations, and analyzing the collected data [82], [83] and applications of embedded systems to collect more personal, participatory data [84].

6.1.2 Future Study

During this study, we discovered several different topics and questions for future studies that would be interesting to test. They are listed here.

1. How the time and the way coaches helped the teams reflects the team's design outcome?
2. How much team member helped each other? From the pictures, it was clear that teams were asking each other's help during the challenge. How much the other team members helped and how much it helped, would be good questions for this kind of a study
3. How much effect does PaperBot challenge have on the learning in ME310 project?
4. How was the code written by the participants evolved during the different stages of the challenge and can this be reflected to measure what the participants learned during the challenge?
5. How the code differs between the teams and can it be reflected the design outcome?
6. PaperBot challenge is a working and fun hands-on design challenge, but could it be made better through games and gamification?

7. Can the learning be accelerated by changing the fundamental structure of the teaching? Would it be better to teach more about coding logic and structures before building the game?
8. Should the teaching about coding be differently organized for different disciplines?

Our goal is to continue developing further the script for automated picture analyzing and methods for image data, time-tracking data and coach note collection methods by fixing the faults found during this study. Adding machine learning to the code would make the script more reliable and robust.

Acknowledgements

It would not have been possible to complete this study without the help of the teaching team involved. Thank you, Diogo Ferreira Pinto, Harri Toivonen, Jussi Hannula, Joona Kurikka, Lauri Repokari, Marko Nieminen, Marko Takala, Markus Nordberg, Peter Tapio and Tuuli Utriainen for all the help. Special thanks to all the amazing people at CERN who spent their time with us, helped us and were giving us precious advice.

Thank you for the students Almeida Pereira Pedro Miguel, Barradas Fontes Cláudia Sofia, Carneiro Tomas, Cavalletti Linda, Costa Beatriz, Costa Fernandes João Andre, Costa Maria Ines, Costa Sofia, De Sousa Neves Moisés, Duarte Soares Manuel Alberto, Ferrari Jacopo, Ferreira Otilia, Gamboa Teixeira da Mota Tomás, Gentilini Francesco, Klemetsen Daniel, Lilleberg Petterson Jonas, Loureiro Sá Ferreira Nuno Filipe, Marimba da Costa José Alberto, Møretrø Tobias, Nascimento Gois Katarina, Oggioni Lucia, Oliveira Couto Diogo Miguel, Oliveira Jorge Neto Pedro Diogo, Pääkkönen Paavo, Padua de Faria Pedro João, Repokari Natalia, Rønnes Maja, Rosati Francesca, Silva Ferreira Ana Laura, Summatavet Märt , Tõke Timmu, Trindade Fonseca Ana , Vagos Gomes Portovedo Lousa Rita and Virki Tarmo for taking part to this study as participants.

References

[1] A. Turing, “Computer Machinery and Intelligence,” *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[2] B. Baron and L. Darling-Hammond, “Prospects and Challenges for Inquiry-based Approaches to Learning,” *Nat. Learn. Using Res. to Inspire Pract.*, pp. 199–225, 2010.

[3] J. Thomas, “A Review of Research on Project-based Learning,” *The Autodesk Foundation*, no. March, 2000.

[4] M. Clavert and M. Laakso, “Implementing Design-dased Learning in Engineering Education: Case Aalto University Design Factory,” in *41st SEFI Conference*, 2013, no. September, pp. 16–20.

[5] B. Trilling and C. Fadel, *21st Century Skills: Learning for Life in Our Times*, 1st ed. Jossey-Bass, 2009.

[6] C. Dym, A. Agogino, Ö. Eris, D. Frey, and L. Leifer, “Engineering Design Thinking, Teaching, and Learning,” *J. Eng. Educ.*, no. January, pp. 103–120, 2005.

[7] I. Teixeira and J. Teixeira, “Challenges of Engineering Education in a Global World,” *Eng. Educ. (CISPEE), 2013 1st Int. Conf. Port. Soc.*, pp. 1–7, 2013.

[8] A. Dutson, R. Todd, S. Magleby, and C. Sorensen, “A Review of Literature on Teaching Engineering Design Through Project-oriented Capstone Courses,” *J. Eng. Educ.*, vol. 86, no. January, pp. 17–28, 1997.

[9] R. Terry and J. Harb, “Kolb, Bloom, Creativity, and Engineering Design,” *ASEE Annu. Conf. Proc.*, vol. 2, pp. 1594–1600, 1993.

[10] B. Hodge and W. Steele, “Experiences with a Curriculum with Balanced Design Content in All Stems,” *ASEE Annu. Conf. Proc.*, vol. 1, pp. 225–211, 1995.

[11] K. Wood, D. Jensen, J. Bezdek, and K. Otto, “Reverse Engineering and Redesign: Courses to Incrementally and Systematically Teach Design,” *J. Eng. Educ.*, vol. 90, no. 3, pp. 363–373, 2001.

[12] M. Regan and S. Sheppard, “Interactive Multimedia Courseware and the Hands-On Learning Experience: An Assessment Study,” *J. Eng. Educ.*, vol. 85, no. 2, pp. 123–131, 1996.

[13] S. D. Sheppard, “Mechanical Dissection : an Experience in How Things Work,” *Proc. Eng. Educ. Conf. Curric. Innov. Integr.*, pp. 1–8, 1992.

[14] S. Sheppard and R. Jennison, “Examples of Freshman Design Education,” *Int. J. Eng. Educ.*, vol. 13, no. 4, pp. 248–261, 1997.

[15] K. Smith, S. Sheppard, D. Johnson, and R. Johnson, “Pedagogies of Engagement: Classroom-based Practices,” *J. Eng. Educ.*, vol. 94, no. January, pp. 87–101, 2005.

[16] H. Kroes and D. Bax, “Domain-independent Descriptive Design Model and Its Application to Structured Reflection on Design Processes,” *Res. Eng. Des.*, pp. 1–36, 2006.

[17] K. Ulrich and S. Eppinger, *Product Design and Development*. 2012.

[18] K. Otto and K. Wood, *Product Design - Techniques in Reverse Engineering and New Product Development*. Prentice Hall, 2001.

[19] L. Leifer and M. Steinert, “Dancing with Ambiguity: Causality Behavior, Design Thinking, and Triple-loop-learning.,” *Inf. Knowl. Syst. Manag.*, vol. 10, no. 1, pp. 151–173, 2011.

[20] V. Krishnan and K. Ulrich, “Product Development Decisions: A Review of the Literature,” *Manage. Sci.*, vol. 47, no. 1, pp. 1–21, 2001.

[21] L. Hassi and M. Laakso, “Design Thinking in the Management Discourse: Defining the Elements of the Concept,” in *18th IPDM Conference*, 2011, pp. 1–14.
[22] L. Kimbell, “Beyond Design Thinking : Design-as-Practice and Designs-in-Practice,” in *CRESC Conference*, 2009, no. May.
[23] U. Johansson-Skoldberg, J. Woodilla, and M. Cetinkaya, “Design Thinking: Past, Present and Possible Futures,” *Creat. Innov. Manag.*, vol. 22, no. 2, pp. 121–146, 2013.
[24] B. Orthel, “Implications of Design Thinking for Teaching , Learning , and Inquiry,” *J. Inter. Des.*, vol. 40, no. 3, pp. 1–20, 2015.
[25] K. Thoring and R. Müller, “Understanding the Creative Mechanisms of Design Thinking : An Evolutionary Approach,” in *Second Conference on Creativity and Innovation in Design*, 2011, no. October, pp. 137 – 147.
[26] T. Brown, “Design Thinking,” *Harvard Business Review*, 2008.
[27] R. Coyne, “Wicked Problems Revisited,” *Des. Stud.*, vol. 26, no. 1, pp. 5–17, 2005.
[28] R. Buchanan, “Wicked Problems in Design Thinking,” *Des. Issues*, vol. 8, no. 2, pp. 5–21, 1992.
[29] B. Head and J. Alford, “Wicked Problems: Implications for Public Policy and Management,” *Adm. Soc.*, vol. 47, no. 6, pp. 711–739, 2015.
[30] T. Leinonen and E. Durall, “Design Thinking and Collaborative Learning,” *Media Educ. Res. J.*, vol. 21, no. 42, pp. 107–115, 2014.
[31] B. Jobst and C. Meinel, “How Prototyping Helps to Solve Wicked Problems,” in *Design Thinking Research - Building Innovation Eco-Systems*, Springer, 2014, pp. 105–113.
[32] L. Hassi and M. Laakso, “Making Sense of Design Thinking,” *IDBM Pap. vol. 1*, pp. 50–62, 2011.
[33] C. Meinel and L. Leifer, “Design Thinking Research,” in *Design thinking Understand - Improve - Apply*, Springer, 2011, pp. xiii–xxi.
[34] H. Sjöman, “Learning Outcomes Through Global Product Innovation Course in Aalto University,” Aalto University, 2014.
[35] M. Lande and L. Leifer, “Prototyping to Learn: Characterizing Engineering Students’ Prototyping Activities and Prototypes,” *Int. Conf. Eng. Des.*, pp. 507–516, 2009.
[36] V. Taajamaa, S. Kirjavainen, L. Repokari, H. Sjöman, T. Utriainen, and T. Salakoski, “Dancing with Ambiguity Design Thinking in Interdisciplinary Engineering Education,” in *2013 IEEE-Tsinghua International Design Management Symposium: Design-Driven Business Innovation, TIDMS 2013*, 2014, pp. 353–360.
[37] T. Carleton and L. Leifer, “Stanford’s ME310 Course as an Evolution of Engineering Design,” in *19th CIRP Design Conference – Competitive Design*, 2009, no. March, pp. 547–554.
[38] ME310 Stanford Teaching Team, “ME310 Design Innovation Booklet.” 2013.
[39] Y. Reich, G. Ullmann, M. Van Der Loos, and L. Leifer, “Coaching Product Development Teams: A Conceptual Foundation for Empirical Studies,” *Res. Eng. Des.*, vol. 19, no. 4, pp. 205–222, 2009.
[40] Y. Reich, G. Ullmann, M. Van Der Loos, and L. Leifer, “Perceptions of Coaching in Product Development Teams,” in *International Conference on Engineering Design, ICED’07*, 2007, no. August.
[41] R. Hackman and R. Wageman, “A Theory Coaching of Team,” *Acad. Manag. Rev.*, vol. 30, no. 2, pp. 269–287, 2005.

[42] Ö. Eris and L. Leifer, “Facilitating Product Development Knowledge Acquisition: Interaction Between the Expert and the Team,” *Int. J. Eng. Educ.*, vol. 19, no. 1, pp. 142–152, 2003.
[43] W. N. C. Dictionaries, *Webster’s New World College Dictionary*, 5th ed. 2014.
[44] V. Thurmond, “The Point of Triangulation,” *J. Nurs. Scholarsh.*, vol. 33, no. 3, pp. 253–258, 2001.
[45] N. Denzin, *Sociological Methods: A Sourcebook*, 5th ed. Aldine, 2006.
[46] J. Kimchi, B. Polivka, and J. Stevenson, “Triangulation: Operational Definitions,” *Nurs. Res.*, vol. 40, no. 6, pp. 362–366, 1991.
[47] S. Prince, *Computer vision: Models, Learning and Inference*. Cambridge University, 2012.
[48] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
[49] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. O’Reilly, 2008.
[50] R. Szeliski, *Computer Vision : Algorithms and Applications*. Springer, 2010.
[51] R. Gonzales and R. Woods, *Digital Image Processing*, 2nd ed. Prentice-Hall, 2002.
[52] Google, “Google Self-driving Car Project.” [Online]. Available: https://www.google.com/selfdrivingcar/. [Accessed: 21-Apr-2016].
[53] ZenRobotics Ltd., “Robotic Waste Separator.” [Online]. Available: http://zenrobotics.com/. [Accessed: 21-Apr-2016].
[54] T. Kemppainen and A. Visala, “Stereo Vision Based Tree Planting Spot Detection,” in *IEEE International Conference on Robotics and Automation*, 2013, pp. 739–745.
[55] OpenCV, “Open Source Computer Vision.” [Online]. Available: http://opencv.org/about.html. [Accessed: 20-Apr-2016].
[56] J. Martin and D. Gill, “The Relationships Among Competitive Orientation, Sport-confidence, Self-efficacy, Anxiety, and Performance,” *J. Sport. Exerc. Psychol.*, vol. 13, pp. 149–159, 1991.
[57] P. Totterdell, “Catching Moods and Hitting Runs: Mood Linkage and Subjective Performance in Professional Sport Teams,” *J. Appl. Psychol.*, vol. 85, no. 6, pp. 848–859, 2000.
[58] R. Oser, A. G. McCallum, E. Salas, and B. B. J. Morgan, “Toward a Definition of Teamwork: An Analysis of Critical Team Behaviors,” no. March, pp. 1–43, 1989.
[59] A. Glickman, S. Zimmer, C. Montero, P. Guerette, W. Campbell, B. Morgan, and E. Salas, “The Evolution of Teamwork Skills: An Empirical Assessment with Implications for Training,” no. November, p. 126, 1987.
[60] B. Morgan, A. Glickman, E. Woodard, A. Blaiwes, and E. Salas, “Measurement of Team Behaviors in a Navy Training Environment,” no. November, p. 106, 1986.
[61] R. Sethi, D. Smith, and W. Park, “Cross-functional Product Development Teams, Creativity, and the Innovativeness of New Consumer Products,” *J. Mark. Res.*, vol. 38, no. 1, pp. 73–85, 2001.
[62] J. Shah, S. Kulkarni, and N. Vargas-Hernandez, “Evaluation of Idea Generation Methods for Conceptual Design: Effectiveness Metrics and Design of Experiments,” *J. Mech. Des.*, vol. 122, no. 4, pp. 377–384, 2000.
[63] P. Skogstad, M. Steinert, K. Gumerlock, and L. Leifer, “We Need a Universal Design Project Outcome Performance Measurement Metric: a Discussion Based on Empirical Research,” in *International Conference on Engineering Design, ICED’09*, 2009, no. August, pp. 473–484.

[64] K. Crow, “Product Development Metrics,” 2001. [Online]. Available: http://www.npd-solutions.com/pdmetrics.html. [Accessed: 26-Apr-2016].
[65] M. Jung and L. Leifer, “A Method to Study Affective Dynamics and Performance in Engineering Design Teams,” in *Internation Conference on Engineering Design, ICED’11*, 2011, no. August.
[66] G. Kress, M. Schar, and M. Steinert, “A Standardized Measurement Tool for Evaluating and Comparing Team Reframing Capabilities,” in *International Design Conference, DESIGN 2012*, 2012, vol. May, pp. 513–522.
[67] C. Redelinghuys and T. Bahill, “A Framework for the Assessment of the Creativity of Product Design Teams,” *J. Eng. Des.*, vol. 17, no. 2, pp. 121–141, 2006.
[68] S. Kavadias and S. Sommer, “The Effects of Problem Structure and Team Diversity on Brainstorming Effectiveness,” *Manage. Sci.*, vol. 55, no. 12, pp. 1899–1913, 2009.
[69] K. E. Soderquist and K. Kostopoulos, “Factors Affecting the Performance of New Product Development Teams: Some European Evidence,” in *Knowledge Perspectives of New Product Development*, 2012, pp. 29–48.
[70] J. Shah and N. Vargas-Hernandez, “Metrics for Measuring Ideation Effectiveness,” *Des. Stud.*, vol. 24, no. 2, pp. 111–134, 2003.
[71] R. Reagans, E. Zuckerman, and B. McEvily, “How to Make the Team: Social Networks vs. Demography as Criteria for Designing Effective Teams,” *Adm. Sci. Q.*, vol. 49, no. 1, pp. 101–133, 2004.
[72] G. Kress and M. Schar, “Teamology – The Art and Science of Design Team Formation,” in *Design Thinking Research - Understanding Innovation*, vol. 36, 2012, pp. 189–209.
[73] G. Kress and M. Schar, “Initial Conditions: The Structure and Composition of Effective Design Teams,” in *International Conference on Engineering Design, ICED’11*, 2011, no. August.
[74] M. Lande, “Work in Progress: Making Room: Creating Design Spaces for Design Practice,” in *Proceedings - Frontiers in Education Conference, FIE*, 2012, pp. 1–5.
[75] IdeaSquare, “In IdeaQsuare, Heaven Is Only 10 Steps Away.” [Online]. Available: http://knowledgetransfer.web.cern.ch/sites/knowledgetransfer.web.cern.ch/files/10 Theses.pdf. [Accessed: 28-Apr-2016].
[76] Gridvision Engineering, “Gleeo Time-tracking Software,” 2016. [Online]. Available: https://gleeo.com/index.php/en/. [Accessed: 02-May-2016].
[77] Adafruit, “Adafruit NeoPixel Digital RGB LED Strip - White 60 LED - WHITE.” [Online]. Available: https://www.adafruit.com/products/1138. [Accessed: 20-Apr-2016].
[78] PJRC, “Teensy USB Development Board.” [Online]. Available: https://www.pjrc.com/teensy/index.html. [Accessed: 20-Apr-2016].
[79] Arduino, “Arduino Nano USB Development Board.” [Online]. Available: http://www.arduino.cc/en/Main/ArduinoBoardNano. [Accessed: 20-Apr-2016].
[80] Arduino, “Arduino UNO USB Development Board.” [Online]. Available: http://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed: 20-Apr-2016].
[81] Arduino, “Arduino Software.” [Online]. Available: http://www.arduino.cc/en/Main/ArduinoBoardNano. [Accessed: 20-Apr-2016].
[82] K. Carrizosa, Ö. Eris, A. Milne, and A. Mabogunje, “Building the Design Observatory: A Core Instrument for Design Research,” in *International Design Conference - Design 2002*, 2002, pp. 37–42.

[83] P. Torlind, N. Sonalkar, M. Bergström, E. Blanco, B. Hicks, and H. Mcalpine, “Lessons Learned and Future Challenges for Design Observatory Research,” in *International Conference on Engineering Design, Iced09*, 2009, pp. 371–382.

[84] C. Kriesi, M. Steinert, L. Aalto-Setälä, A. Anvik, S. Balters, A. Baracchi, M. Bisballe Jensen, L. E. Bjørkli, N. Buzzaccaro, D. Cortesi, F. D’Onghia, C. Dosi, G. Franchini, M. Fuchs, A. Gerstenberg, E. Hansen, K. Hiekkanen, D. Hyde, I. Ituarte, J. Kalasniemi, J. Kurikka, I. Lanza, A. Laurila, T. H. Lee, S. Lønvik, A. Mansikka-Aho, M. Nordberg, P. Oinonen, L. Pedrelli, A. Pekuri, E. Rane, T. Reime, L. Repokari, M. Rønningen, S. Rowlands, H. Sjöman, K. Slåttsveen, A. Strachan, K. Strømstad, S. Suren, P. Tapio, T. Utriainen, M. Vignoli, S. Vijaykumar, T. Welo, and A. Wulvik, “Distributed Experiments in Design Sciences, a Next Step in Design Observation Studies?,” in *International Conference on Engineering Design, ICED15*, 2015, no. July, pp. 319–328.

Appendixes

Appendix 1: Script for arranging files to folders according to timestamp. 2 pages.

Appendix 2: Script for copying and renaming camera 5 images. 1 page.

Appendix 3: Script for copying every second file from a folder. 1 page.

Appendix 4: Script for copying wanted files to one location. 1 page.

Appendix 5: Code for defining the mask range limits. 4 pages.

Appendix 6: Script for finding wanted color from an image. 4 pages.

Appendix 7: Script for flipping images 180 degrees horizontally. 3 pages.

Appendix 8: Script for collecting the timestamp from specific files. 4 pages.

Appendix 9: Code for defining the color of a certain pixel. 1 page.

Appendix 10: Script for representing an image as a number grid. 3 pages.

Appendix 1: Script for arranging files to folders according to timestamp. 2 pages.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import os
import sys
import shutil
from datetime import datetime, date, time

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Script for arranging files to folder according to the file's
 modification timestamp
'''

mon = 0
tue = 0
wed = 0
thu = 0
fri = 0
off = 0

walk_dir = sys.argv[1]

print('\nwalk_dir = ' + walk_dir)
print('walk_dir (absolute) = ' + os.path.abspath(walk_dir))

for root, subdirs, files in os.walk(walk_dir):
    print('--\nroot = ' + root)
    for subdir in sorted(subdirs):
        print('\t- subdirectory ' + subdir)

    for filename in sorted(files):
        file_path = os.path.join(root, filename)

        print('\t- file %s (full path: %s)' % (filename, file_path))

        # Get the modification date and time from files
        t = os.path.getmtime(file_path)
        dt = datetime.fromtimestamp(t)
        tt = datetime.timetuple(dt)
        month = tt[1]
        day = tt[2]
        time = tt[3] * 100 + tt[4]
        print('\t\t-Day: {}\tTime: {}'.format(day, time))

        if day == 11 and month == 1:
            if time >= 1930 and time <= 2210:
                shutil.copy2(file_path, './MONDAY/')
                mon += 1
                print('\t\t-copy {} to ./MONDAY/'.format(filename))
        elif day == 12 and month == 1:
            if time >= 1045 and time <= 2300:
```

```
61                  shutil.copy2(file_path, './TUESDAY/')
62                  tue += 1
63                  print('\t\t-copy {} to ./TUESDAY/'.format(filename))
64          elif day == 13 and month == 1:
65              if time >= 1000 and time <= 2300:
66                  shutil.copy2(file_path, './WEDNESDAY/')
67                  wed += 1
68                  print('\t\t-copy {} to ./WEDNESDAY/'.format(filename))
69          elif day == 14 and month == 1:
70              if time >= 1000 and time <= 2359:
71                  shutil.copy2(file_path, './THURSDAY/')
72                  thu += 1
73                  print('\t\t-copy {} to ./THURSDAY/'.format(filename))
74          elif day == 15 and month == 1:
75              if time >= 0 and time <= 59:
76                  shutil.copy2(file_path, './THURSDAY/')
77                  thu += 1
78                  print('\t\t-copy {} to ./THURSDAY/'.format(filename))
79              if time >= 100 and time <= 1700:
80                  shutil.copy2(file_path, './FRIDAY/')
81                  fri += 1
82                  print('\t\t-copy {} to ./FRIDAY/'.format(filename))
83          else:
84              print('\t\t-{}, date out of scope'.format(filename))
85              off += 1
86
87  print('\nMonday:\t\t{}\nTuesday:\t\t{}\nWednesday:\t{}\nThursday:\t{} \
88          \nFriday:\t\t{}\nOff scope:\t{}'.format(mon, tue, wed, thu, \
89          fri, off))
90
91  print('')
92
```

Appendix 2: Script for copying and renaming camera 5 images. 1 page.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import os
import sys
import shutil
import fnmatch

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */
'''
 Script for gathering wanted files from several subfolders into one
 folder and giving the files a unique filename
'''

walk_dir = sys.argv[1]
dst = './IMAGES/'

print('\nwalk_dir = ' + walk_dir)
print('walk_dir (absolute) = ' + os.path.abspath(walk_dir))

for root, subdirs, files in os.walk(walk_dir):
    print('--\nroot = ' + root)
    for subdir in sorted(subdirs):
        print('\t- subdirectory ' + subdir)

    for filename in sorted(files):
        file_path = os.path.join(root, filename)
        print('\t- file %s (full path: %s)' % (filename, file_path))

        if fnmatch.fnmatch(file_path, '*100GOPRO*'):
            prefix = '100_'
        elif fnmatch.fnmatch(file_path, '*101GOPRO*'):
            prefix = '101_'
        elif fnmatch.fnmatch(file_path, '*102GOPRO*'):
            prefix = '102_'
        elif fnmatch.fnmatch(file_path, '*103GOPRO*'):
            prefix = '103_'
        elif fnmatch.fnmatch(file_path, '*104GOPRO*'):
            prefix = '104_'
        elif fnmatch.fnmatch(file_path, '*105GOPRO*'):
            prefix = '105_'
        elif fnmatch.fnmatch(file_path, '*106GOPRO*'):
            prefix = '106_'
        elif fnmatch.fnmatch(file_path, '*107GOPRO*'):
            prefix = '107_'
        else:
            prefix = 'xxx_'

        new_filename = prefix + filename
        shutil.copy2(file_path, dst + new_filename)
        print('\t\tNew filename:{}\n'.format(new_filename))
print('')

```

Appendix 3: Script for copying every second file from a folder. 1 page.

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-

from __future__ import print_function
import os
import sys
import shutil

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Script to copy every second file in a folder and subfolders to a
 specified location. Original file folder is given as an argument
'''

walk_dir = sys.argv[1]
dst = './every_second_image_from_original/'
i = 0

print('\nwalk_dir = ' + walk_dir)
print('walk_dir (absolute) = ' + os.path.abspath(walk_dir))

for root, subdirs, files in os.walk(walk_dir):
    print('--\nroot = ' + root)
    for subdir in sorted(subdirs):
        print('\t- subdirectory ' + subdir)

    for filename in sorted(files):
        file_path = os.path.join(root, filename)

        print('\t- file %s (full path: %s)' % (filename, file_path))

        if i%2 == 0:
            print('\t\t- Copy file {} to {}'.format(filename, dst))
            shutil.copy2(file_path, dst + filename)

        i += 1

print('')
```

Appendix 4: Script for copying wanted files to one location. 1 page.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import os
import sys
import shutil
import fnmatch

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Script to gather the wanted pictures to one place.
 Folder given as argument
'''

walk_dir = sys.argv[1]
dst = './IMAGES/'

print('\nwalk_dir = ' + walk_dir)
print('walk_dir (absolute) = ' + os.path.abspath(walk_dir))

for root, subdirs, files in os.walk(walk_dir):
    print('--\nroot = ' + root)
    for subdir in sorted(subdirs):
        print('\t- subdirectory ' + subdir)

    for filename in sorted(files):
        file_path = os.path.join(root, filename)

        print('\t- file %s (full path: %s)' % (filename, file_path))

        if fnmatch.fnmatch(filename, 'M*.jpg'):
            prefix = root.replace('/', '_')
            new_filename = prefix + '_'+ filename
            shutil.copy2(file_path, dst + new_filename)
        else:
            continue

print('')

```

Appendix 5: Code for defining the mask range limits. 4 pages.

```
#!/usr/bin/python

import sys
import os
import glob as g
import numpy as np
import cv2

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Code for defining the range limits for the mask.

 Control keys:
 q -> l_h + 1
 Q -> l_h - 1
 a -> l_s + 1
 A -> l_s - 1
 z -> l_v + 1
 Z -> l_v - 1

 w -> u_h + 1
 W -> u_h - 1
 s -> u_s + 1
 S -> u_s - 1
 x -> u_v + 1
 X -> u_v - 1
'''

# Initial values
l_h = 139
l_s = 122
l_v = 190
u_h = 179
u_s = 168
u_v = 209

def waitForInput( key ):
    global l_h
    global l_s
    global l_v
    global u_h
    global u_s
    global u_v
    valid_key = True
    any_key = False

    # If key is more than one char, make sure that it is capitalized
    if len(key) > 1:
        key.lower()
        key.capitalize()

    if len(key) == 1:
```

```
61              key = ord(key)
62          elif key == 'Esc':
63              key = 27
64          elif key == 'Any':
65              any_key = True
66          else:
67              print 'Can\'t recognise key \'{}\''.format(key)
68              valid_key = False
69
70          while(valid_key):
71              if any_key:
72                  cv2.waitKey(0)
73                  break
74              else:
75                  read = cv2.waitKey(5) & 0xFF
76                  if read == key:
77                      shutdown()
78                      break
79                  elif read == ord('q'):
80                      if l_h < 180:
81                          l_h += 1
82                          break
83                          #printLimits()
84                  elif read == ord('Q'):
85                      if l_h > 0:
86                          l_h -= 1
87                          break
88                          #printLimits()
89                  elif read == ord('a'):
90                      if l_s < 255:
91                          l_s += 1
92                          break
93                          #printLimits()
94                  elif read == ord('A'):
95                      if l_s > 0:
96                          l_s -= 1
97                          break
98                          #printLimits()
99                  elif read == ord('z'):
100                     if l_v < 255:
101                         l_v += 1
102                         break
103                         #printLimits()
104                 elif read == ord('Z'):
105                     if l_v > 0:
106                         l_v -= 1
107                         break
108                         #printLimits()
109
110                 elif read == ord('w'):
111                     if u_h < 180:
112                         u_h += 1
113                         break
114                         #printLimits()
115                 elif read == ord('W'):
116                     if u_h > 0:
117                         u_h -= 1
118                         break
119                         #printLimits()
120                 elif read == ord('s'):
```

```
121                     if u_s < 255:
122                         u_s += 1
123                         break
124                         #printLimits()
125                 elif read == ord('S'):
126                     if u_s > 0:
127                         u_s -= 1
128                         break
129                         #printLimits()
130                 elif read == ord('x'):
131                     if u_v < 255:
132                         u_v += 1
133                         break
134                         #printLimits()
135                 elif read == ord('X'):
136                     if u_v > 0:
137                         u_v -= 1
138                         break
139                         #printLimits()
140
141
142 def readImage( img ):
143     print '\nReading image:\t\t{}'.format(img)
144     return cv2.imread(img)
145
146 def cropImage( img, x1, x2, y1, y2 ):
147     return img[y1:y2, x1:x2]
148
149 def showImage( name, img ):
150     height, width, channel = img.shape
151     print 'Show image:\t{}\t({}px\t{}x{}px)'.format(name, img.size, \
152             width, height)
153     cv2.namedWindow(name,cv2.WINDOW_NORMAL)
154     cv2.imshow(name,img)
155     return True
156
157 def blur( img ):
158     # Blur the origin
159     print '***Blurring the image'
160     return cv2.blur(img,(20,20))
161
162 def convertToHSV( img ):
163     # Make a HSV version of the origin
164     print '***Converting BGR to HSV'
165     return cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
166
167 def buildMask( img ):
168     # Define range of wanted color in HSV
169     #lower_limit = np.array([4,189,152])
170     #upper_limit = np.array([11,255,255])
171
172     lower_limit = np.array([l_h,l_s,l_v])
173     upper_limit = np.array([u_h,u_s,u_v])
174
175     # Threshold the HSV image to get only wanted colors
176     print '***Finding the wanted color'
177     mask = cv2.inRange(img, lower_limit, upper_limit)
178     #return clearNoise(mask)
179     return mask
180
```

```
181 def clearNoise( img ):
182     # Clean noise from the mask and fill small holes
183     print '***Clearing out noise'
184     kernel = np.ones((10,10),np.uint8)
185     new_image = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
186     new_image = cv2.morphologyEx(new_image, cv2.MORPH_CLOSE, kernel)
187     return new_image
188 
189 def applyMask( mask, img ):
190     # Bitwise-AND mask and original image
191     print '***Apply mask to the original image'
192     return cv2.bitwise_and(img,img, mask= mask)
193 
194 def printLimits():
195     print '\nLower limits:\t{}\t{}\t{}'.format(l_h,l_s,l_v)
196     print 'Upper limits:\t{}\t{}\t{}'.format(u_h,u_s,u_v)
197 
198 def shutdown():
199     printLimits()
200     print '\nShutting down...\n\n'
201     cv2.destroyAllWindows()
202     quit()
203 
204 def main():
205     f = 'image.jpg'
206     # Read image from the filelist
207     image = readImage(f)
208 
209     cropped = cropImage(image, x1 = 1200, x2 = 3500, y1 = 950, y2 =3024)
210 
211     # Blur the image for better results
212     blurred = blur(image)
213     # Convert the image From BGR to HSV
214     hsv = convertToHSV(blurred)
215     '''
216     # Build the mask (find the beanies with threshold)
217     mask = buildMask(hsv)
218     # Apply mask to original image
219     result = applyMask(mask, image)
220     '''
221     # Show some image
222     showImage('Original',image)
223 
224     #waitForInput('Esc')
225 
226     while(1):
227         printLimits()
228         # Build the mask (find the beanies with threshold)
229         mask = buildMask(hsv)
230         result = applyMask(mask, image)
231         #show mask
232         #showImage('Result',result)
233         showImage('Cropped',cropped)
234         waitForInput('Esc')
235 
236     shutdown()
237     pass
238 
239 if __name__=="__main__":
240     sys.exit(main())
```

Appendix 6: Script for finding wanted color from an image. 4 pages.

```
#! /usr/bin/env python2

import sys
import os
import glob as g
import numpy as np
import cv2

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Script for finding the wanted color (beanie this time) from the
 picture. NOTICE: This script is not finished and is missing critical
 features and functions
'''

def checkArgument( argv ):
    if len(argv) == 1:
        print '\nGive directory as argument\n'
        quit()
    elif len(argv) > 2:
        print '\nGive ONLY one argument\n'
        quit()
    elif len(argv) == 2:
        if os.path.isdir(argv[1]):
            print '\n'
            return
        else:
            print '\nArgument not directory\n'
            quit()

def waitForInput( key ):
    valid_key = True
    any_key = False

    # If key is more than one char, make sure that it is capitalized
    if len(key) > 1:
        key.lower()
        key.capitalize()

    if len(key) == 1:
        key = ord(key)
    elif key == 'Esc':
        key = 27
    elif key == 'Any':
        any_key = True
    else:
        print 'Can\'t recognise key \'{}\''.format(key)
        valid_key = False

    print '\nPress \'{}\' to continue'.format(key)

    while(valid_key):
```

```
61          if any_key:
62              cv2.waitKey(0)
63              break
64          else:
65              read = cv2.waitKey(10) & 0xFF
66              if read == key:
67                  break
68
69  def listFilenames( directory ):
70      return g.glob(directory)
71
72  def readImageDateTime( img ):
73      datetime = ''
74      year = 0
75      month = 0
76      day = 0
77      hours = 0
78      minutes = 0
79      seconds = 0
80
81      f = open(img, 'rb')
82
83      # Return Exif tags
84      tags = exifread.process_file(f)
85      f.close()
86
87      for tag in tags:
88          if tag in ('Image DateTime', 'EXIF DateTimeDigitized'):
89              datetime = str(tags[tag])
90              year = int(datetime[0:4])
91              month = int(datetime[5:7])
92              day = int(datetime[8:10])
93              hours = int(datetime[11:13])
94              minutes = int(datetime[14:16])
95              seconds = int(datetime[17:19])
96              print '{}.{}.{}\t{}:{}:{}'.format(day, month, year, \
97                      hours, minutes, seconds)
98              return year, month, day, hours, minutes, seconds
99      return None
100
101
102 def readImage( img ):
103     print '\nReading image:\t\t{}'.format(img)
104     return cv2.imread(img)
105
106 def showImage( name, img ):
107     print 'Show image:\t\t{}\t({}px)'.format(name, img.size)
108     cv2.namedWindow(name,cv2.WINDOW_NORMAL)
109     cv2.imshow(name,img)
110     return True
111
112 def blur( img ):
113     # Blur the origin
114     print '***Blurring the image'
115     return cv2.blur(img,(20,20))
116
117 def convertToHSV( img ):
118     # Make a HSV version of the origin
119     print '***Converting BGR to HSV'
120     return cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
```

```
121
122 def buildMask( img ):
123     # Define range of wanted color in HSV
124     lower_limit = np.array([4,189,152])
125     upper_limit = np.array([11,255,255])
126
127     # Threshold the HSV image to get only wanted colors
128     print '***Finding the wanted color'
129     mask = cv2.inRange(img, lower_limit, upper_limit)
130     return clearNoise(mask)
131
132 def clearNoise( img ):
133     # Clean noise from the mask
134     print '***Clearing out noise'
135     kernel = np.ones((10,10),np.uint8)
136     new_image= cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
137     new_image= cv2.morphologyEx(new_image, cv2.MORPH_CLOSE, kernel)
138     return new_image
139
140
141 def applyMask( mask, img ):
142     # Bitwise-AND mask and original image
143     print '***Apply mask to the original image'
144     return cv2.bitwise_and(img,img, mask= mask)
145
146 def findContours( img ):
147     # Find possible contours from image
148     print '***Finding countours'
149     contours, hierarchy = cv2.findContours(img,cv2.RETR_TREE, \
150             cv2.CHAIN_APPROX_NONE)
151     return contours
152
153 def shutdown():
154     print '\nShutting down...\n\n'
155     cv2.destroyAllWindows()
156     quit()
157
158 def main(argv):
159     window_open = False
160     beanies = dict()
161
162     # Check that the argument is ok
163     checkArgument(argv)
164
165     print 'Reading files from:\t' + argv[1]
166     directory = argv[1] + '*'
167
168     # Read all the files from the given directory
169     filelist = listFilenames(directory)
170     print '{} file(s) found'.format(len(filelist))
171
172     # Do your magic for all the images
173     for f in filelist:
174         contours = []
175
176         # Read image from the filelist
177         image = readImage(f)
178         # Read Date and Time of the picture
179         readImageDateTime( f )
180         # Blur the image for better results
```

```
181             blurred = blur(image)
182             # Convert the image From BGR to HSV
183             hsv = convertToHSV(blurred)
184             # Build the mask (find the beanies with threshold)
185             mask = buildMask(hsv)
186             # Apply mask to original image
187             result = applyMask(mask, image)
188 
189             # Show some image
190             window_open = showImage('1',image)
191             window_open = showImage('2',mask)
192 
193             # Use the built mask to find contours
194             temp_contours = findContours( mask )
195             if len(temp_contours) == 0:
196                 print 'No contours found'
197             else:
198                 print '{} contour(s) found\n'.format(len(temp_contours))
199                 for c in temp_contours:
200                     c_area = cv2.contourArea(c)
201                     (c_x,c_y),c_r = cv2.minEnclosingCircle(c)
202 
203                     # If the size of the contour is over 30 px
204                     if c_area > 50.0:
205                         contours.append(c)
206 
207                     print '{}\t{}\t{}\t{}'.format(int(c_area), int(c_x), \
208                             int(c_y), int(c_r))
209 
210             # If we found one or more beanies
211             # Add to the dictionary {name: number of beanies found}
212             if len(contours) > 0:
213                 beanies[f] = len(contours)
214 
215             # If we opened a window to show an image, wait for user input
216             if window_open:
217                 waitForInput('Any')
218 
219 
220     print '\n{} images(s) found'.format(len(beanies))
221     print beanies
222 
223     shutdown()
224     pass
225 
226 if __name__=="__main__":
227     sys.exit(main(sys.argv))
228 
```

Appendix 7: Script for flipping images 180 degrees horizontally. 3 pages.

```
#!/usr/bin/python

import sys
import os
import glob as g
import numpy as np
import cv2

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

folder = './Flipped_images/'

'''
 Script to turn the picture 180 degrees. NOTICE: the saved file will
 have the file datetime set for the current day
 Folder as argument
'''
def checkArgument( argv ):
    if len(argv) == 1:
        print '\nGive directory as argument\n'
        quit()
    elif len(argv) > 2:
        print '\nGive ONLY one argument\n'
        quit()
    elif len(argv) == 2:
        if os.path.isdir(argv[1]):
            print '\n'
            return
        else:
            print '\nArgument not directory\n'
            quit()

def waitForInput( key ):
    valid_key = True
    any_key = False

    # If key is more than one char, make sure that it is capitalized
    if len(key) > 1:
        key.lower()
        key.capitalize()

    if len(key) == 1:
        key = ord(key)
    elif key == 'Esc':
        key = 27
    elif key == 'Any':
        any_key = True
    else:
        print 'Can\'t recognise key \'{}\''.format(key)
        valid_key = False

    while(valid_key):
        if any_key:
```

```
61                  cv2.waitKey(0)
62                  break
63              else:
64                  read = cv2.waitKey(5) & 0xFF
65                  if read == key:
66                      shutdown()
67                      break
68
69
70  def listFilenames( directory ):
71      return g.glob(directory)
72
73  def readImage( img ):
74      print '\nReading image:\t\t{}'.format(img)
75      return cv2.imread(img)
76
77  def showImage( name, img ):
78      height, width, channel = img.shape
79      print 'Show image:\t{}\t({}px\t{}x{}px)'.format(name, img.size, \
80              width, height)
81      cv2.namedWindow(name,cv2.WINDOW_NORMAL)
82      cv2.imshow(name,img)
83      return True
84
85  def saveImage( img, img_name ):
86      # write the cropped image to disk
87      print 'Saving image:\t\t{}'.format(img_name)
88      cv2.imwrite(img_name, img)
89
90  def horizontalFlip( img ):
91      return cv2.flip(img,0)
92
93  def shutdown():
94      printLimits()
95      print '\nShutting down...\n\n'
96      cv2.destroyAllWindows()
97      quit()
98
99  def main(argv):
100
101     # Check that the argument is ok
102     checkArgument(argv)
103
104     print 'Reading files from:\t' + argv[1]
105     directory = argv[1] + '*'
106
107     # Read all the files from the given directory
108     filelist = listFilenames(directory)
109     print '{} file(s) found'.format(len(filelist))
110
111     i = 1
112     # Do your magic for all the images
113     for f in filelist:
114         filepath = str(f)
115         name = filepath[9::]
116         # Read image from the filelist
117         image = readImage(f)
118         # Read Date and Time of the picture
119         readImageDateTime(f)
120         # Flip the image 180 degrees
```

```
121             flip = horizontalFlip(image)
122
123             #showImage('original', image)
124             #showImage(name, flip)
125
126             # Save the image to specific folder
127             saveImage(flip, folder + name)
128
129             #waitForInput('Esc')
130
131         shutdown()
132         pass
133
134     if __name__=="__main__":
135         sys.exit(main(sys.argv))
136
```

Appendix 8: Script for collecting the timestamp from specific files. 4 pages.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from __future__ import print_function
import os
import sys
import shutil
import fnmatch
from datetime import datetime, date, time

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Script that will go through a folder of pictures and searches for the
 ones listed in the arrays below. Also makes a copy of each wanted
 picture to corresponding file.
'''

walk_dir = sys.argv[1]
dst_team_in = './TEAM_IN/'
dst_J_in = './COACH1_IN/'
dst_M_in = './COACH2_IN/'
dst_P_in = './COACH3_IN/'
dst_other_in = './OTHER_IN/'

in_array = ['']
out_array = ['']
J_in = ['']
J_out = ['']
M_in = ['']
M_out = ['']
P_in =['']
P_out =['']
other_in = ['']
other_out = ['']

team_index = 0
J_index = 0
M_index = 0
P_index = 0
other_index = 0

stop_program = False

team_name = 'Curium'
all_images_csv_file = open('all_images.csv', 'w')
csv_file = open('images.csv', 'w')

#Check that the arrays are equal length, if not stop script
if len(in_array) != len(out_array):
    print('len(in_array) != len(out_array)')
    stop_program = True
if len(J_in) != len(J_out):
```

```
61      print('len(J_in) != len(J_out)')
62      stop_program = True
63  if len(M_in) != len(M_out):
64      print('len(M_in) != len(M_out)')
65      stop_program = True
66  if len(P_in) != len(P_out):
67      print('len(P_in) != len(P_out)')
68      stop_program = True
69  if len(other_in) != len(other_out):
70      print('len(P_in) != len(P_out)')
71      stop_program = True
72  if stop_program:
73      exit()
74
75  print('Filename,Array index,Team name,Action,Weekday,Timestamp, \
76      Date,Time', file=csv_file)
77  print('Filename,Team name,Weekday,Timestamp,Date,Time', \
78      file=all_images_csv_file)
79
80  print('\nwalk_dir = ' + walk_dir)
81  print('walk_dir (absolute) = ' + os.path.abspath(walk_dir))
82
83  #Check what team pictures script is going through
84  if fnmatch.fnmatch(os.path.abspath(walk_dir), '*/1/*'):
85      team_name = 'Nobelium'
86  elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/2/*'):
87      team_name = 'Neptumium'
88  elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/3/*'):
89      team_name = 'Actinium'
90  elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/4/*'):
91      team_name = 'Thorium'
92  elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/5_1/*'):
93      team_name = 'Fermium'
94  elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/5_2/*'):
95      team_name = 'Lemmium'
96  elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/6/*'):
97      team_name = 'Curium'
98  elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/7/*'):
99      team_name = 'Einsteinium'
100 elif fnmatch.fnmatch(os.path.abspath(walk_dir), '*/8/*'):
101     team_name = 'Americium'
102
103 for root, subdirs, files in os.walk(walk_dir):
104     print('--\nroot = ' + root)
105
106     for subdir in sorted(subdirs):
107         print('\t- subdirectory ' + subdir)
108
109
110     for filename in sorted(files):
111         file_path = os.path.join(root, filename)
112         print('\t- file %s (full path: %s)' % (filename, file_path))
113
114         t = os.path.getmtime(file_path)
115         dt = datetime.fromtimestamp(t)
116         tt = datetime.timetuple(dt)
117         d = '{}.{}.{}'.format(tt[2],tt[1],tt[0])
118         t = '{}:{}:{}'.format(tt[3],tt[4],tt[5])
119         timestamp = '{} {}'.format(d,t)
120         weekday = dt.strftime("%A")
```

```
121
122         ####################################################
123         # Print to all_images.csv
124         csv = '{0},{1},{2},{3},{4},{5}'.format(filename,team_name, \
125                 weekday,timestamp,d,t)
126         print(csv, file=all_images_csv_file)
127
128
129         action = ''
130         image_name = filename[0:8]
131
132         ####################################################
133         # in_array & out_array
134         if image_name in in_array:
135             team_index = in_array.index(image_name)
136             action = 'team_in'
137             csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
138                     team_index,team_name,action,weekday,timestamp,d,t)
139             print(csv, file=csv_file)
140
141         elif image_name in out_array:
142             team_index = out_array.index(image_name)
143             action = 'team_out'
144             csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
145                     team_index,team_name,action,weekday,timestamp,d,t)
146             print(csv, file=csv_file)
147
148         if image_name >= in_array[team_index] and \
149                 image_name <= out_array[team_index]:
150             shutil.copy2(file_path, dst_team_in)
151             print('\t\t- Copy picture {} to {}'.format(filename, \
152                     dst_team_in))
153
154         ####################################################
155         # J_in & J_out
156         if image_name in J_in:
157             J_index = J_in.index(image_name)
158             action = 'coach1_in'
159             csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
160                     J_index,team_name,action,weekday,timestamp,d,t)
161             print(csv, file=csv_file)
162
163         elif image_name in J_out:
164             J_index = J_out.index(image_name)
165             action = 'coach1_out'
166             csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
167                     J_index,team_name,action,weekday,timestamp,d,t)
168             print(csv, file=csv_file)
169
170         if image_name >= J_in[J_index] and image_name <= J_out[J_index]:
171             shutil.copy2(file_path, dst_coach1_in)
172             print('\t\t- Copy picture {} to {}'.format(filename, \
173                     dst_coach1_in))
174
175         ####################################################
176         # M_in & M_out
177         if image_name in M_in:
178             M_index = M_in.index(image_name)
179             action = 'coach2_in'
180             csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
```

```
                    M_index,team_name,action,weekday,timestamp,d,t)
            print(csv, file=csv_file)

        elif image_name in M_out:
            M_index = M_out.index(image_name)
            action = 'coach2_out'
            csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
                    M_index,team_name,action,weekday,timestamp,d,t)
            print(csv, file=csv_file)

        if image_name >= M_in[M_index] and image_name <= M_out[M_index]:
            shutil.copy2(file_path, dst_coach2_in)
            print('\t\t- Copy picture {} to {}'.format(filename, \
                    dst_coach2_in))

        ##################################################
        # P_in & P_out
        if image_name in P_in:
            P_index =P_in.index(image_name)
            action = 'coach3_in'
            csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
                    P_index,team_name,action,weekday,timestamp,d,t)
            print(csv, file=csv_file)

        elif image_name in P_out:
            P_index = P_out.index(image_name)
            action = 'coach3_out'
            csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
                    P_index,team_name,action,weekday,timestamp,d,t)
            print(csv, file=csv_file)

        if image_name >= P_in[P_index] and image_name <= P_out[P_index]:
            shutil.copy2(file_path, dst_coach3_in)
            print('\t\t- Copy picture {} to {}'.format(filename, \
                    dst_coach3_in))

        ##################################################
        # other_in & other_out
        if image_name in other_in:
            other_index =other_in.index(image_name)
            action = 'other_in'
            csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
                    other_index,team_name,action,weekday,timestamp,d,t)
            print(csv, file=csv_file)

        elif image_name in other_out:
            other_index = other_out.index(image_name)
            action = 'other_out'
            csv = '{0},{1},{2},{3},{4},{5},{6},{7}'.format(filename, \
                    other_index,team_name,action,weekday,timestamp,d,t)
            print(csv, file=csv_file)

        if image_name >= other_in[other_index] and \
                image_name <= other_out[other_index]:
            shutil.copy2(file_path, dst_other_in)
            print('\t\t- Copy picture {} to {}'.format(filename, \
                    dst_other_in))

print('')
```

Appendix 9: Code for defining the color of a certain pixel. 1 page.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy as np
import cv2

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Program to find out what is the color code of a clicked pixel
'''

def on_mouse_click (event, x, y, flags, frame):
    if event == cv2.EVENT_LBUTTONUP:
        clr = img[x,y]
        #print clr
        bgr = np.uint8([[clr]])
        print ('BGR:\t{}'.format(bgr))
        hsv = cv2.cvtColor(bgr,cv2.COLOR_BGR2HSV)
        print ('HSV:\t{}'.format(hsv))
        print ''
        #hsv = cv2.cvtColor(bgr,cv2.COLOR_BGR2HSV_FULL)
        #print hsv



# Read and show the image, name the window and mouse callback
img = cv2.imread('image.jpg')
cv2.namedWindow('image', cv2.WINDOW_NORMAL)
cv2.setMouseCallback('image', on_mouse_click)
cv2.imshow('image', img)


while(1):
    if cv2.waitKey(5) & 0xFF == 27:
        break
cv2.destroyAllWindows()

```

Appendix 10: Script for representing an image as a number grid. 3 pages.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from __future__ import print_function
import numpy as np
import cv2

#/* Copyright (C) 2016 Jani Kalasniemi - All Rights Reserved
# * You may use, distribute and modify this code under the
# * terms of the Creative Commons Attribution-NonCommercial
# * 4.0 International License. To view a copy of this license,
# * visit: http://creativecommons.org/licenses/by-nc/4.0/
# *
# * OR write to: jani.kalasniemi@iki.fi
# */

'''
 Script for presenting an image as a number grid based on the color
 of the pixel
'''
# Change here the image you want to present as color values
image_source = 'bw_eye.jpg'

# True  = black & white mode
# False = colored mode
B_W = False

'''
 Simplify the colored pixel presentation or select only one of the
 channels
'''
# Simplify will calculate average between the channel color values
# Blue prints out only blue value
# Green prints out only green value
# Red prints out only red value
simplify = False
blue = False
green = False
red = False

# Read the image black and white
if B_W:
    img = cv2.imread(image_source, 0)
else:
    img = cv2.imread(image_source)

# Get the width and length of the picture
if B_W:
    height, width = img.shape
else:
    height, width, channels = img.shape

print("\n") #Just to give the answer some more space

if B_W and simplify or blue or green or red:
    print("Cannot present Black & White values together with Average, \
           Blue, Green or Red values\n")

if B_W:
    print("Black & White values")
```

```
61  elif simplify:
62      print("Averaged channels values")
63  elif blue:
64      print("Blue channel values")
65  elif green:
66      print("Green channel values")
67  elif red:
68      print("Red channel values")
69  else:
70      print("BGR values (The LONG and MESSY presentation)")
71
72  if B_W:
73      print("width: {}\theight: {}\n".format(width, height))
74  else:
75      print("width: {}\theight: {}\tchannels: {}\n".format(width, \
76              height, channels))
77
78  for y in range(0,height):
79      for x in range(0,width):
80
81          if simplify and not B_W:
82              values = [img[x,y,0], img[x,y,1], img[x,y,2]]
83              px = int(np.mean(values))
84          elif blue and not B_W:
85              px = img[x,y,0]
86          elif green and not B_W:
87              px = img[x,y,1]
88          elif red and not B_W:
89              px = img[x,y,2]
90          else:
91              px = img[x,y]
92
93          if B_W or simplify or blue or green or red:
94              if px < 10:
95                  e = "   "
96              elif px < 100:
97                  e = "  "
98              else:
99                  e = " "
100         else:
101             if px[0] < 10:
102                 blue_end = "   "
103             elif px[0] < 100:
104                 blue_end = "  "
105             else:
106                 blue_end = " "
107             if px[1] < 10:
108                 green_end = "   "
109             elif px[1] < 100:
110                 green_end = "  "
111             else:
112                 green_end = " "
113             if px[2] < 10:
114                 red_end = "   "
115             elif px[2] < 100:
116                 red_end = "  "
117             else:
118                 red_end = " "
119
120             e = blue_end + green_end + red_end + " "
```

Appendix 10 (3/3)

```
121
122             print(px, end=e)
123         print("")
124
125 print("\n") #Just to give the answer some more space
126
127
```