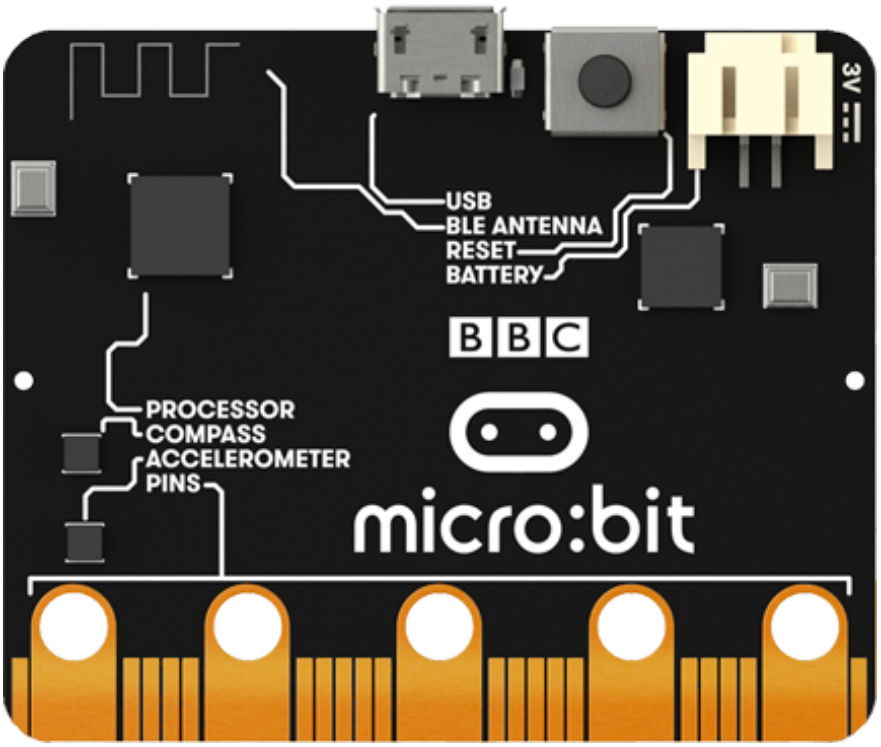
CHALMERS | UNIVERSITY OF GOTHENBURG



Introducing micro:bit in Swedish primary schools

An empirical design research on developing teaching material for training computational thinking in Swedish primary schools

Master's thesis in Interaction Design

Niklas Carlborg, Marcus Tyrén

Department of Applied IT
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2016:175

Introducing micro:bit in Swedish primary schools

An empirical design research on developing teaching material for training computational thinking in Swedish primary schools

NIKLAS CARLBORG, MARCUS TYRÉN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Applied IT
Interaction Design and Technologies
Chalmers University of Technology
Gothenburg, Sweden 2017

Introducing micro:bit in Swedish primary schools
An empirical design research on developing teaching material for training computational thinking in Swedish primary schools
NIKLAS CARLBORG, MARCUS TYRÉN

Supervisor: Eva Eriksson, Interaction Design and Technologies
Examiner: Staffan Björk, Interaction Design and Technologies

Master's Thesis 2016:175
Department of Applied IT
Interaction Design and Technologies
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Backside of BBC micro:bit hardware.

Introducing micro:bit in Swedish primary schools
An empirical design research on developing teaching material for training computational thinking in Swedish primary schools
NIKLAS CARLBORG, MARCUS TYRÉN
Department of Applied IT
Chalmers University of Technology

Abstract

During the 21st century there has been an increasing interest in the field of computational thinking, a popular way of teaching students about programming. In a society with an ever faster technical development it becomes more relevant to educate future generations about the technology that surrounds us. Many different platforms can be used for this purpose, e.g Scratch, Raspberry Pi or Arduino. In the UK the platform micro:bit has been used in schools since 2016. Other countries are now also incorporating programming in their curriculum, and Sweden is set to incorporate this by the 1st of July 2018. This thesis examines what may be important to consider when designing teaching materials with the micro:bit for training Swedish primary school students' computational thinking skills. This was done through an iterative design process, by conducting 21 workshops with the goal to support Swedish primary school teachers with micro:bit teaching materials. The result of this thesis consists of 9 individual parts, presented in 4 groups, mapped along an axis of abstraction. A model was created in an attempt to communicate observed relationships between students learning potential, their risk of feeling overwhelmed and the amount of choices they were provided with. A set of guidelines as well as a teaching approach was provided to give more concrete answers to the research question. Practical workshop examples were also provided in an attempt to aid teachers in the transition to the new curriculum.

Keywords: micro:bit, teaching material, programming, primary school, computational thinking.

Acknowledgements

We would like to thank Carl Heath and Peter Ljungstrand at RISE Interactive for giving us the opportunity for this thesis and also for their help and support during the work. We would also like to thank our supervisor, Eva Eriksson, for her guidance and perseverance and also everyone that participated in our workshops, both students and teachers.

Contents

Glossary

exercise A task designed to enable someone to gain certain knowledge or develop certain skills. 88

learner Someone acquiring a skill or knowledge. In some theories learners are differentiated from students, in this thesis however, students are also considered to be learners. 20

microcontroller A small computer on a circuit board that can be programmed. 7

motivation Something energizing and directing behaviours and activities. 17

platform An environment in which a piece of software can be executed. v

STEM Science, technology, engineering, and mathematics. 7, 12

student A person enrolled in a school. 2

teaching material Some content, tool or practice designed for teachers, students and or self-learners to aid in the acquiring of certain knowledge or skills. v

URL Uniform Resource Locator, also known as a web-address. 9

workshop A set of exercises that follow a predetermined plan for the lesson in order to reach a set goal. All participants are present on site. 31

1

Introduction

In a society with accelerating technical development, there are obvious difficulties in designing education that can prepare next generations for an unknown future. With an ever faster technical development it becomes less relevant to teach specific skills that might become obsolete in a near future, instead it becomes more important to teach skills that enable new generations to swiftly adapt to the changes and technologies that emerge. The National Research Council address this in their report "Being Fluent with Information Technology" stating that the rapid technology development calls for a 'fluency' approach, rather than the traditional 'skill-based' approach (1; 2). There is a central challenge in learning how to work with technology and design through iterative, reflective and flexible approaches to learning(3).

Sweden is set to introduce programming in schools 1st of july 2018. Principals will be able to choose when to apply the changes within a one year period, starting july 1st 2017(4).There has been little research done on how the teacher's technological skills and attitudes towards technology will affect such a transition nor on what school resources will be required. The current research has focused on impediments that arise regarding the shift in mindset that is required from teachers in a more explorative teaching setting, rather than the more traditional goal oriented approach(3).

In the UK the transition to including programming in education has already begun. As part of the Make It Digital initiative in 2015, BBC has together with Microsoft, Samsung and other partners, developed the micro:bit for use in computer education. Every year 7 pupil in the UK was given one of these small computers, or microcontrollers, that can be programmed and customized. It aims to inspire young people to get creative in the digital world, developing core skills in STEM subjects and produce a new generation of inventors and makers.

The Swedish research institute RISE Interactive is also exploring ways to use the micro:bit as a teaching platform. They are inspired by the UK and want to see how this platform can be adapted to aid in the Swedish curriculum transition. This thesis aims to investigate this endeavour in order to support Swedish teachers in the transition to the new programming curriculum.

1.1 Purpose

This master's thesis aims to answer an academic research question through the pursuit of a design goal. Hence this aspiration has one foot in the academic world and another foot in the field of applied interaction design. The outcome will therefore both be an academic contribution as well as a potential product or service deliverable.

1.1.1 Research Question

What is important to consider when designing teaching materials with the BBC micro:bit for training Swedish primary school students computational thinking skills?

1.1.1.1 Contribution

This thesis aims to contribute a set of guidelines to consider when introducing a new technological platform, such as the BBC micro:bit, for training computational thinking in Swedish primary schools. These guidelines will be based on non exhaustive empirical design research and be limited to the micro:bit platform and the Swedish school context.

1.1.2 Design Goal

Support Swedish primary school teachers with teaching materials, based on the BBC micro:bit, that help them meet the programming requirements of the new curriculum changes.

1.1.2.1 Deliverables

To support Swedish teachers in the transition to a programming curriculum, this thesis aims to deliver teaching materials that meet the needs of the teachers. Due to the explorative design approach used in this project, it is hard to define the outcoming nature of such support at the beginning of the project.

1.2 Stakeholders

Stakeholders are defined as people that either affect or are affected by the system that is being designed for. The stakeholders that are within the scope of this thesis are presented below.

1.2.1 Students (user)

The focus of this master's thesis is on primary school students spanning from 4th to 6th grade in public Swedish schools. As the British micro:bit efforts have been targeted at students of this age, it is assumed that this is an appropriate age. Another

assumption is that these students are sufficiently knowledgeable in the english language to be able to work with the micro:bit. They are assumed to be familiar with consumer technology but unfamiliar with programming tools and digital fabrication.

1.2.2 Teachers (user)

Another focus is the primary school teachers, who teach 4th-6th grades in a variety of subjects in public Swedish schools. Teachers being on the brink of a big change in their field of work. They have to acclimatize quickly to meet the new curriculum changes, support students in their development as new technologies are introduced while conforming to limited school resources. Teachers may differ significantly in age and education as well as attitudes towards technology and change.

1.2.3 RISE Interactive (business client)

RISE Interactive (TII) is a Swedish governmentally owned research institute part of the Swedish ICT and Rise concern, working with industrial research and innovation globally. Their mission is to courageously do new things in the fields of technology, business, and design that allow people to do and think in new ways through empowering collaborative design. This thesis falls in line with their current project Makerskola which aims to use technology in creative new ways to contribute to the development of subject matter specific methodologies. TII provide the research context, the design problem as well as access to a large network of teachers that have access to an even larger number of students. TII also provide technical, academic and project related knowledge in weekly supervision sessions.

1.2.4 Interaction Design Faculty (academic client)

The master's programme of Interaction Design at Chalmers University of technology are concerned with teaching the skills of designing the interactions between people and products with information technology as a central component. Moreover Chalmers purpose is to carry on education and research on an internationally high level within the fields of engineering, science and mathematics-natural sciences. Chalmers provide the opportunity for this thesis and provide academic and research related knowledge in frequent supervision sessions.

1.2.5 Thesis Authors

The thesis work is performed by two students in their last semester of their master's study in Interaction Design and Technologies at Chalmers. Their backgrounds are bachelor's degrees at Chalmers in Industrial design engineering and Computer Sciences respectively. Their aspirations are to gain insight in the planning, execution and presentation of a larger scale project where they are allowed to exercises previously acquired skills as well as gain first hand experience to better prepare them for a future profession in interaction design.

1.3 Delimitations

Due to the wicked problem nature of the research question and the limited time frame for the project, a non exhaustive empirical design research approach was undertaken. The consequential delimitations are presented below to set the scope for this project.

As there are many factors and stakeholders involved in a nation's educational system, this project was only able to regard a limited scope within the frame of a master's thesis. The system in which school based teaching and learning exist, involves many stakeholders that either affect or are are affected by the system. Potential stakeholders are students, parents, teachers, school managers as well as researchers, policymakers, companies and other organisations. Due to time constraints the main focus of this master's thesis was on the student and teacher stakeholders. The Swedish government and the National Agency of Education are both major political stakeholders, in this thesis however, their curriculum changes will be treated as given limitations for simplification. The governmental curriculum changes affect multiple school subjects, this thesis will however primarily focus on the programming aspects relating to mathematical and technical school subjects. The year groups studied in this project are limited to year 4-6. The societal impact of digitalisation and other topics relating to humanities and social sciences, are outside of the scope of this thesis.

Many different technologies and development platforms are available and new ones are constantly being developed. Choosing what technological platform to invest in might be a very relevant question to educators, however this is not within the scope of this thesis. This project will be using the BBC micro:bit, as a given educational platform as the client stakeholders considered it to be affordable, already well spread in the UK, and having a lot of potential with its many onboard sensors. Furthermore there are multiple different editors available for working with the micro:bit. This thesis chose however to only look at the Microsoft MakeCode micro:bit editor, and specifically the block editor part of it.

2
Background

This chapter presents a more detailed description about the suggested changes in Swedish schools regarding IT-strategies and programming in the classroom and curriculum. An overview is given to the changes that have been made in the UK as well as the resources and methods that are used to help teachers in the transition to a computing curriculum. The selected hardware platform for the project, the micro:bit, is described with its pro's and con's together with related hardware products that are also suited for classroom use.

2.1 Changes in Swedish Education Strategies

Thursday 9th of March 2017 the Swedish government made changes to the policy documents that control Swedish primary and secondary school curriculum(4). The main focus of the changes are to enhance and emphasize the school's duty in strengthening the students digital competences. This is planned to be done throughout the various year groups by different means. Varying from teaching step-wise instructions in the early year groups to fully encompass programming in later year groups. These changes are to be adopted by Swedish schools starting July 1st 2017, and be fully implemented by July 1st 2018. The changes are primarily concerned with:

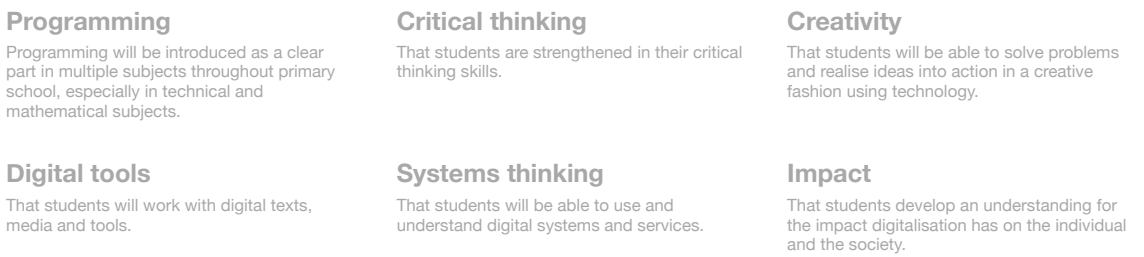


Figure 2.1: An English translation of the new curriculum changes

Some of these changes were previously proposed by the Swedish National Agency for Education. They developed drafts of possible changes to policy documents regarding the mission to propose suggestions to the national it-strategies for the school system(5). These focused on enhancing and clarifying digital competence in the policy documents. Their definition of digital competences was based on the descriptions used by the EU and The Digitalisation Commission.

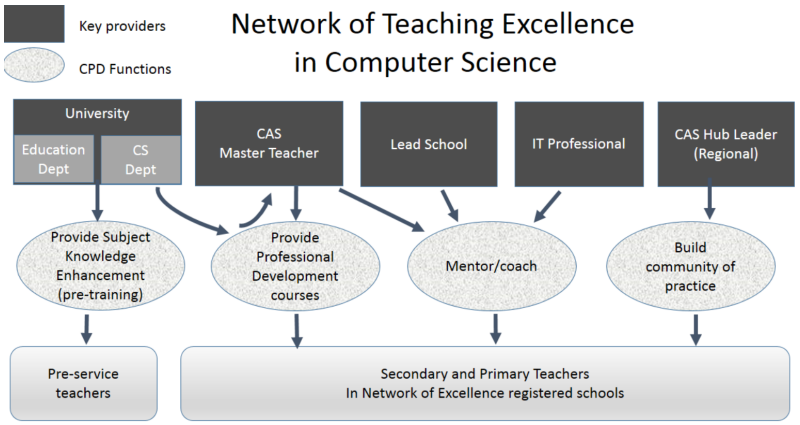
2.2 Programming in UK Education

In the UK the transition to more programming in school started off in January 2012 when The Royal Society published a highly-rated article(6) that put computer science back in the light again with recommendations to reintroduce CS in schools. Up until then they had been teaching information and communication technology (ICT) but with an increasingly declining reputation over the last couple of years. ICT in contrast to CS focused more on the usage and software rather than the creative and underlying principles of computing. It was not long before the department of education declared the ICT curriculum to be rewritten and in its stead officially reintroduce CS teaching in schools again. With this change several issues were brought to surface. For instance how will the primary schools handle the fast pace of these changes, and how will they make sure that there are enough teachers with the right knowledge?

Primary school teachers are the ones that face the most change as their curriculum is the first to be remade while secondary teachers have some more time to prepare for curriculum changes. This is particularly difficult since most primary school teachers are generalists, they teach a whole class in most subjects having broad knowledge rather than being specialists within a certain subject. As there is a massive shortage in teachers with experience and knowledge within the computing area, the delivery of computing in class is a big challenge and might be hampered severely if teachers is not educated fast enough(7). As there is no or very limited resources as well as time being an issue to educate teachers in CS, new methods have to be developed to help teachers in an efficient manner.

Computing At School (CAS) Network of Computer Science Teaching Excellence was started in an attempt to address these issues by building a network that aims to become self-sustained within a three year period. CAS is providing teachers with a system to share resources and host discussion groups through a website. Teachers are allowed to both upload their own and give feedback on others ideas and resources regarding CS in education, this way many teachers feel that they can make a contribution and help their own peers. Additionally CAS Network maintains a number of so called Master CS teachers that are used to deliver professional development to teachers at site. This way a faster paced development of CS skills among teachers is achieved, by teachers teaching teachers the knowledge expands, and will eventually reach a sufficient number of CS educated teachers in the UK.

This is an overview of Network of Teaching Excellence in CS and depicts the roles and areas of involved actors to explain the whole system in addition to the parts explained in previous paragraph.



2.3 About Micro:bit

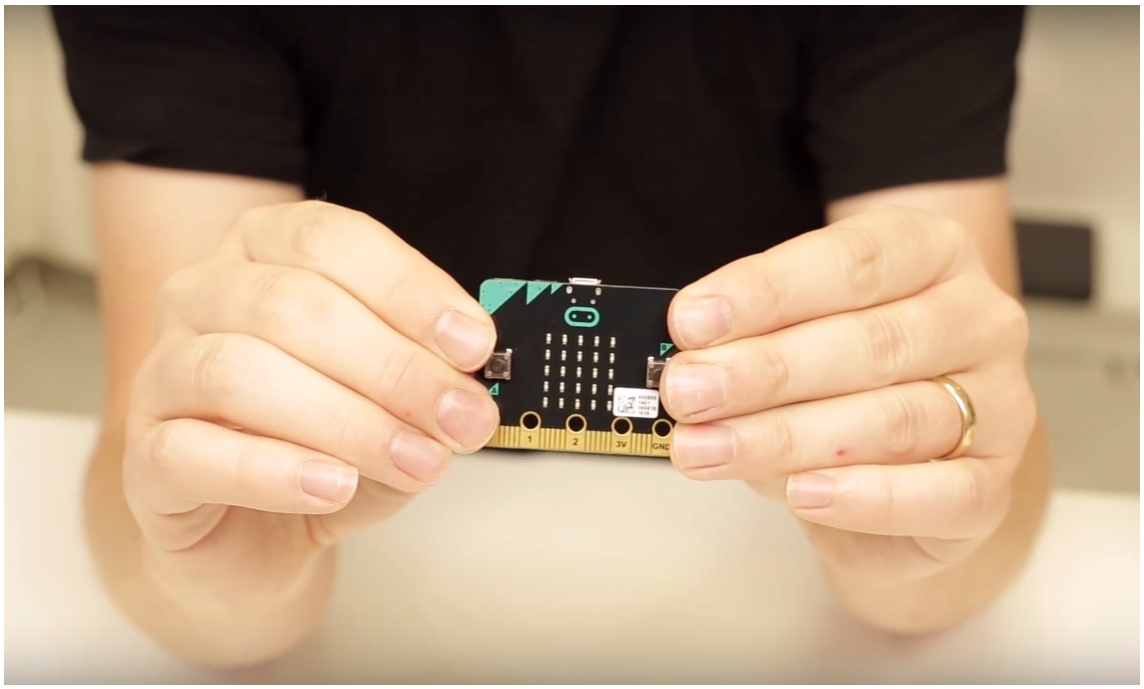


Figure 2.2: The micro:bit hardware held for scale

As a part of BBC's 2015 Make it Digital Initiative the micro:bit was developed. It aims to inspire young people to get creative in the digital world, developing core skills in STEM subjects and produce a new generation of inventors and makers. The micro:bit is a small computer, or microcontroller, that can be programmed and customized in order to bring ideas to life(8). Displaying your name, or making it blink can be coded in seconds even if the user is totally new to programming. micro:bit can also be connected to other devices or sensors and can complement other hardware like Arduino and Raspberry Pi, it works as a great springboard to more complex learning(8). Key features include:

- 5x5 LED matrix display
- Two programmable buttons
- Accelerometer that can detect movement
- A built-in compass to sense direction
- The ability to sense temperature and light levels.
- Bluetooth Smart Technology to interact with other micro:bits and mobile devices.
- Five Input and Output (I/O) rings to connect the micro:bit to devices or sensors using e.g crocodile clips.

For ensuring that the micro:bit becomes successful, it has been mentioned to be important that all partners involved work closely with both teachers, educators and schools to provide resources and information supporting the curriculum.

2.3.1 Editor

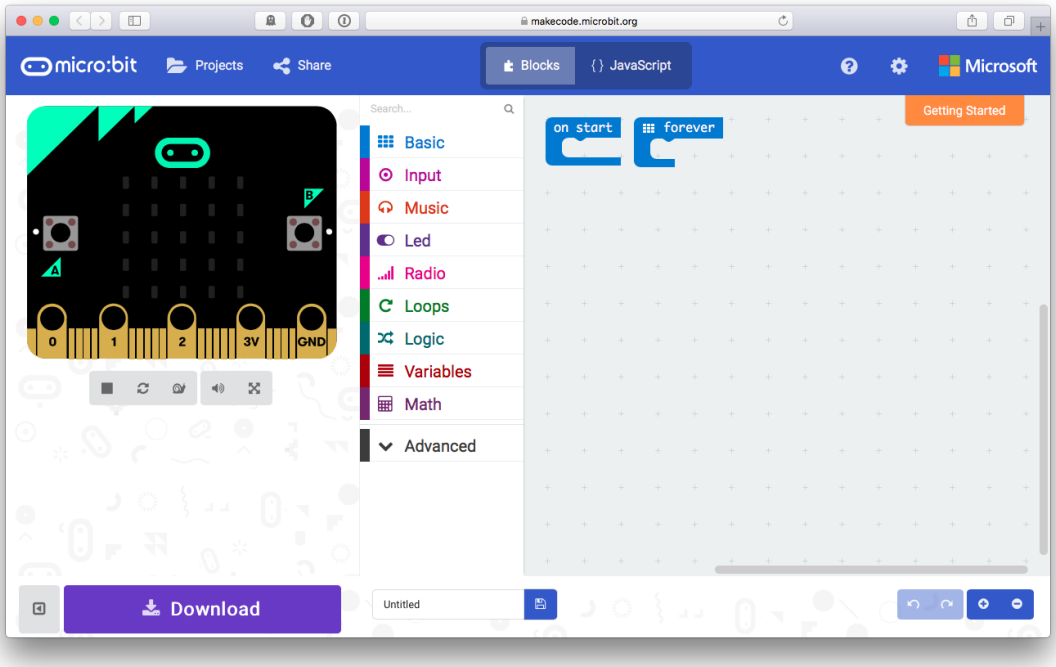


Figure 2.3: One of the editors in which programs can be created for the micro:bit

There are multiple ways of programming the micro:bit, the scope of this thesis has however been limited to only focus on the Microsoft MakeCode micro:bit editor.

The Microsoft MakeCode micro:bit editor is a free to use online JavaScript/Blocks editor for programming the micro:bit. This means that it runs in the web browser and hence is cross platform compatible, both on different web browsers but also across different operating systems, such as OSX, Windows, iOS and Android. This also implies that an internet connection is required for using the editor. Technically

the Microsoft MakeCode editor for micro:bit can be used offline as the application gets cached locally but only if an online compilation has been made first. So either way an internet connection is required at some point.

The term block editor refers to the puzzle like interaction where the user builds their programs by snapping different function blocks together to create a programs behaviour.

The editor allows the user to code both with blocks as well as JavaScript code. It provides the possibility to switch back and forth between these on the fly to translate from one to the other.

The user interface of the editor as seen from left to right consists of a simulator, a section of available blocks and an area where the user is to drag blocks and build their programs. At the top, controls are available for saving and loading projects, switching back and forth between block or JavaScript mode, and some more advanced settings. At the bottom the download button is located for downloading the created program.

For creating a simple program the user begins with finding the desired blocks in the middle column folders and drags them onto the block building area on the right. Blocks snapped into the "on start" block will only run once, and blocks snapped into the "forever" block will repeat indefinitely. The functionality of the program can then be evaluated with the simulator on the left. To download the program the user clicks the download button on the bottom left, and transfers the obtained file to the micro:bit flash drive via a usb cable.

By the time this thesis was written the URL to the editor was:
https://makecode.microbit.org/.

2.4 Related Work

In this chapter three subcategories of related work will be briefly described. Development platforms entail mostly hardware that are related to the micro:bit and how they are used in an educational context. Learning platforms is about different forms of teaching materials on how to learn programming as well as other ways of educating teachers in CS. Lastly a short introduction to maker movements will be presented and how its community is growing and providing schools an alternative way of looking at education with digital materials.

2.4.1 Development Platforms

In this chapter a brief look of existing development platforms, mainly different kinds of hardware products, will be presented. The perspective is from the point of impact in an educational sense and the different uses and environments it is applicable in.

2.4.1.1 Scratch

Scratch is a free programming language that was developed by MIT Media Lab, and has been around since 2013(Scratch 2). Its main purpose is to be accessible for students and teachers and be an easy-to-use tool in introducing computer science through programming, indirectly provide a stepping stone to a world of more advanced programming. As the creation of programs is relatively easy and skills learnt can be used later when learning Java or Python, it works great as an introductory language. Main users are kids around the ages of 9-16 but can be used successfully in classes of both younger and older students(9).

2.4.1.2 Arduino

Arduino is an easy-to-use electronics platform that is used worldwide by makers, students, hobbyists and professionals. Arduino provides open-source on both hardware and software, the boards can read inputs as light sensors and turn it into outputs e.g activating a motor. Arduino was first designed to be an easy tool for quick prototyping aimed at students with little to no background in either programming or electronics, but as it started to reach a wider community the Arduino board changed in order to adapt to new needs and challenges(10).

2.4.1.3 Makey Makey

Makey Makey is an electronic invention tool that lets users connect mundane objects to control computer programs. Makey Makey uses closed loop electrical signals to detect keyboard strokes or mouse click signals by having alligator clips connect between objects and the circuit board. This allows the Makey Makey to work with any computer program or web page, as the inputs are the same(11).

2.4.1.4 Raspberry Pi

The Raspberry Pi is a credit-card sized, single board computer that you can plug into your TV and keyboard. Developed by the Raspberry Pi Foundation to promote teaching computer science at a basic level in schools and developing countries. It is very much like a desktop computer in pocket size and very capable for electronics projects, browsing the web, spreadsheet or playing games and high-definition video(12). A teacher training course, the Picademy, was also started by the Raspberry Pi Foundation aiming to help teachers prepare for the coming additions of computing in the curriculum using the Raspberry Pi, in addition a continuation of the course, professional development, is given free for teachers(13).

2.4.1.5 Quirkbot

Quirkbot is a microcontroller aimed for kids to program and play with. It is a toy that is compatible with Strawbees, another open construction toy, and readily available materials like drinking straws, LEDs and servo motors to create a vast variety of homemade toys. Quirkbot also provides guidance for teachers with their education guide that is filled with both inspirational projects as well as lesson plans(14).

2.4.2 Learning Platforms

A brief look into how teacher material and education can be made accessible. Hour of Code representing the web based program to promote the fun of CS and no prerequisite knowledge of programming is required. CAS is a project in the UK whose aim is to provide a highly accessible network for teacher development in the field of CS, managing master teacher classes, shared online resources and discussion groups.

2.4.2.1 Hour of Code

Hour of Code started of as an hours introduction to CS, with the purpose to play down programming and show that anyone is capable of learning the fundamentals of coding. Today Hour of Code is a worldwide grassroots movement with plenty of guides and activities for all to take part of. Anyone can arrange an Hour of Code with the help of a how-to guide in their school, voluntary or at work. No previous experience is needed as the program has a well defined self-instructed activity for all ages and levels of experience. Most importantly Hour of Code is about having fun and being creative with CS, reaching a wide spectrum of participants with all ages and backgrounds. Teachers also get confidence in successfully teaching a subject they are not educated in, and often spur on further interest in learning CS more deeply(15).

2.4.2.2 Computing at School (CAS)

Computing at School (CAS) is a project funded by the Department of Education in the UK which goal is to help teachers, both primary and secondary, share ideas and resources as well as learn more about how to practice CS in the classroom. CAS consist of a community that has its members run regional hubs around the country where they meet to talk and learn from professionals about teaching CS. The main goal is to equip all involved in computing education with strategic guidance focusing on CS and the computing curriculum, setting a high standard for the level of CS education delivered by those involved(16).

2.4.3 Maker Movements

Maker movement stems from the DIY tradition and foster a learning through-doing setting that focuses on being explorative and curious with technology, mixing both physical and software technologies. Makerspaces is where like-minded ‘makers’ come together in a social environment to have fun and build, as well as share, their projects with others using technology, science, digital art etc.

2.4.3.1 Fab Lab

From MIT’s Center for Bits and Atoms(CBA) comes an educational component, Fab Lab, that is an extension to its research into digital fabrication and computation. Fab Lab works as a prototyping platform for both innovation and invention with digital materials. A local place to come and play, create, learn and invent

with others in a so called 'makerspace'(17). All labs share a common set of tools and processes and form a global network of inventors, spanning over 30 countries around the world. Fab Labs only uses off-the-shelf tools and open source software in order to be available for everyone. Across the countries schools have increased their interest in Fab Labs by using their makerspaces for projects in STEM education. The labs give a very authentic context to operate in, allowing students to design things of pure personal interest and not being tied down by any curriculum. Students can work freely and make use of a proper explorative design process, imagine; design; prototype; reflect; and iterate as they find solutions to their challenges and bring ideas to life. Fab Labs being closely aligned with MIT's CBA, where research into next generation fabrication tools and software are pushing the digital and analog boundaries, make the Fab Labs a cutting edge workshop for research and development.

2.4.3.2 Techshop

TechShop is a community that provides its members access to instruction, professional equipment and software, and a creative space to work in. It works as a DIY workshop and fabrication studio, a local space where entrepreneurs, artists, makers and students learn and work alongside each other. People of all skill levels join in to build on their own projects. Currently TechShop is only available on nine different locations in the United States(18).

2.4.3.3 Makerskola

Makerskola, or Makerspace in schools, is a project supported by Sweden's innovation agency Vinnova. With creative use of emerging technologies their goal is to make a contribution in the development of new subject matter specific methodology. By letting young people explore the boundary between analog and digital resources providing a test in both theoretical and practical work. Over time the project has the intention to improve schools' educational activities in general and provide input for curriculum development, but also provide opportunities to develop and spread the best practices in the field of maker culture between teachers, schools and local education authorities. Many research institutes, businesses and about 30 local education authorities are involved in the project. In order to evaluate methods, equipment and logistics, several testbeds have been established. This gives possibilities for teachers, together with students, to explore the idea of makerspaces in schools; introduction of programming; and creative work with Internet of Things. Once a year a conference is organised, Maker Days, to inspire and share knowledge to which stakeholders outside the partnership are welcome to participate. This shows the projects aim to also be about emphasizing human resource development(19).

2.4.3.4 Digitalverkstan

Digitalverkstan is an investment from Dataföreningen to try and stimulate and develop children's digital knowledge, both in school and leisure. Dataföreningen is a non-profit and unreliant association that work toward a positive development of the

possibilities technology provide in today's society. Digitalverkstan consists of several different programs with different activities that creates opportunities for children's up to fifteen years old to create digitally and practice programming. They provide a service to schools to host workshops of different kinds on various locations around the Gothenburg area. To give them an opportunity to be shown the many possibilities of programming and digital creation based on platforms like Scratch, Arduino, Makey-makey and micro:bit. To facilitate the workshops they hire students that seek a developing and inspiring job on the side(20).

3

Theory

In the following section descriptions of theories that are relevant to the project will be presented. The theories selected were weighed and picked with relevance to the field and own approach in addition to the limited time for the whole project in mind.

3.1 Teacher's Role in Digital Fabrication

As digital fabrication technologies makes increasing impact on supporting STEM subjects in primary and secondary school, the teacher's role to handle these new learning processes of both technology and design has been largely overlooked. There are many challenges that is presented to teachers by introducing digital fabrication in technology in an educational environment, Smith et al(3) have identified four impediments that have to be solved in order to create a healthy environment for teachers and teaching when it comes to integrating technology to support education with the teacher's role in focus. To begin with the schools today are more or less goal-oriented, this is due to classes following a strict curriculum and need to satisfy certain objectives. In order to introduce digital fabrication technologies to support education successfully there has to be a change regarding the curriculum that gives the teacher a different kind of role. Design processes is more of an open ended and explorative way of learning, the learners should be allowed more freedom during class and teacher's role should change towards a facilitator(3). There needs to be a way for teachers to practice purposeful education and still be able to support the explorative process that is digital fabrication.

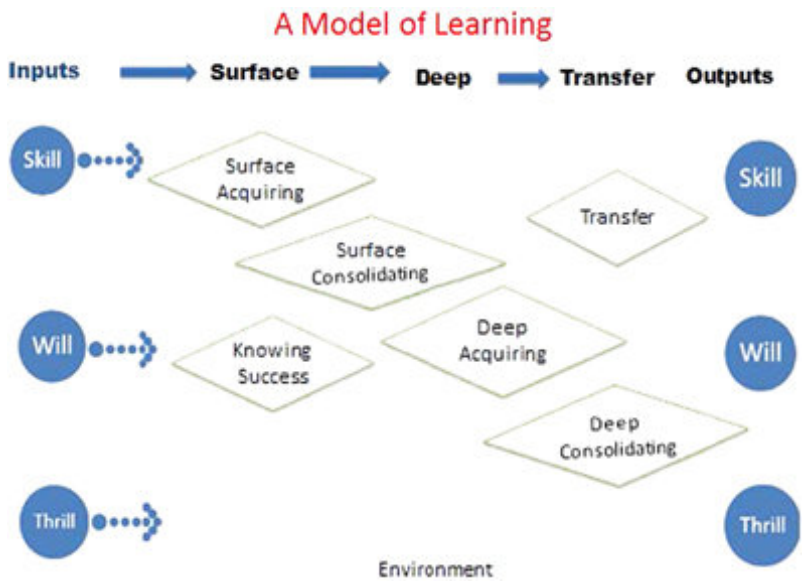
In today's goal-oriented school environment a lot of the teacher's focus is on completion of tasks, the process of getting to a finalization is not as important. When designing with digital materials a big part of the learning process and understanding is through sketching a solution, reflecting and iterating to reach a possible solution. A change in the mindset of looking at design materials and fabrication materials as reflection tools rather than just outcomes of a design process can be a contributing factor to be able to integrate digital fabrication in the classroom. Closely tied is the need for a design language, a common ground of understanding between teachers and students to express ideas and qualities regarding design(3). Teachers must develop this way of reflective understanding, as it is a fundamental part of digital fabrication and design. Another thing to bear in mind is that this will probably rewrite the map of the regular classroom teaching ways. Teachers will not always be in full control of steering a class with precision each time, but rather has to

get accustomed to having less authority and less control due to not mastering all the techniques that are taught. This is a scary situation for any teacher , as the current classroom situation differ greatly from the self-motivated environments as makerspaces are, which need a teacher in a facilitator role(3).

3.2 Constructionism

Seymour Papert(21) built constructionism on the idea of constructivism, that knowledge is a structure built in the mind of the learner rather than something prepackaged ready to be absorbed from the teacher. Constructionism however also adds the notion that the learner constructs this knowledge while consciously creating some public entity, whether it is a sand castle or a theory of the universe. Papert stresses the irony in trying to come up with a definition for constructionism, as the whole idea about it, is that the knowledge about it is created by you as you engage in an effort to create it.
Papert claims that comparing constructionism to instructionism is trying to compare something that is different on a much deeper level than merely the way in which knowledge is acquired, but rather on the level of what the nature of knowledge really is. Further he illustrates the successful implementation of constructionism in stories about children who are exposed to an environment in which their desire to create something beautiful leads them to wanting to learn the math knowledge, for instance, required to implement these ideas(21).

3.3 Hattie and Donoghue Model of Learning



Hattie and Donoghue propose a model of learning(22) that suggests learning has three inputs and outputs: skill, will and thrill. It mentions the importance of defining the success criteria to the learner and that there are three phases of learning: surface,

deep and transfer. Surface and deep are also each divided into an acquiring and a consolidation phase. The model suggests that some learning strategies are more effective than others but that this is dependent on the learning phase. Further on it is argued that learning strategies should be embedded into subject content rather than be taught separately out of context. Transfer is shown to be highly effective for learning, especially in looking at similarities and differences between different contexts and situations. It is suggested that transfer requires the previous phases to be passed in a linear fashion. However, the authors also point out that this is an assumption and that more research needs to look at how the order of phases impact learning.

3.4 Self-Determination Theory

Self-Determination Theory is a macro theory of human motivation that first and foremost states that motivation has more dimensions than simply the strength amount of motivation. According to SDT the type or quality of the motivation is even more important than the strength, for being able to predict psychological effects relating to well-being, performance and creativity.

In SDT motivation, contrasted to amotivation, is described as energizing and directing behaviours and activities. These motivations are divided into the two distinct groups of autonomous motivation and controlled motivation. Autonomous motivation consists both of intrinsic motivation and extrinsic motivations where people have identified with the value of a certain activity or even integrated it into their sense of self. According to Ryan and Deci(23) intrinsic motivation is where one is moved to act for the inherent satisfaction of doing the activity, not driven by any outcome separate from the activity itself. Whereas extrinsic motivation is described as an activity instrumental to reach an outcome separate from the activity at hand. Controlled motivation on the other hand is described as extrinsic motivation that either is external motivation regulation or introjected regulation. Here external regulations are either rewards or punishments whereas introjected regulations are approval motive, avoidance of shame, contingent self-esteem or ego-involvements.

Central to SDT are the notions from Basic Psychological Needs Theory that psychological well-being and performance are predictable on three basic needs: autonomy, competence, and relatedness. To the extent of which these three needs are satisfied or thwarted by the context an individual's differences are changed in two ways, according to SDT. These two individual differences are: causality orientations and aspirations. Causality orientations are derived from Causality Orientations Theory and relates to three ways of orienting oneself in relation to regulating one's behaviours. These orientations are: autonomy, which is acting out of interest; controlled, which is focused on rewards, approval and gains; and impersonal or amotivated, which is an anxious relation to competence. According to SDT all individuals have degrees of all three orientations. These are suggested to be influenced by an individual's surroundings support for the three basic needs (autonomy, competence, and relatedness) and have been shown to correlate with a person's psychological

and behavioural outcome(24). Aspirations or life goals are, according to SDT, goals acquired by an individual to compensate for thwarted basic psychological needs (autonomy, competence, and relatedness) over time. These goals are either intrinsic aspirations or extrinsic aspirations. Intrinsic aspirations are for instance affiliation, generativity, and personal development. Extrinsic affiliations include goals such as fame, wealth and attractiveness. It is suggested that thwarted basic psychological needs result in the adoption of extrinsic life goals in an effort to try to satisfy these needs, something that extrinsic goals are unable to satisfy. At the same time the aspiration for external life goals tend to crowd out basic need satisfaction(24).

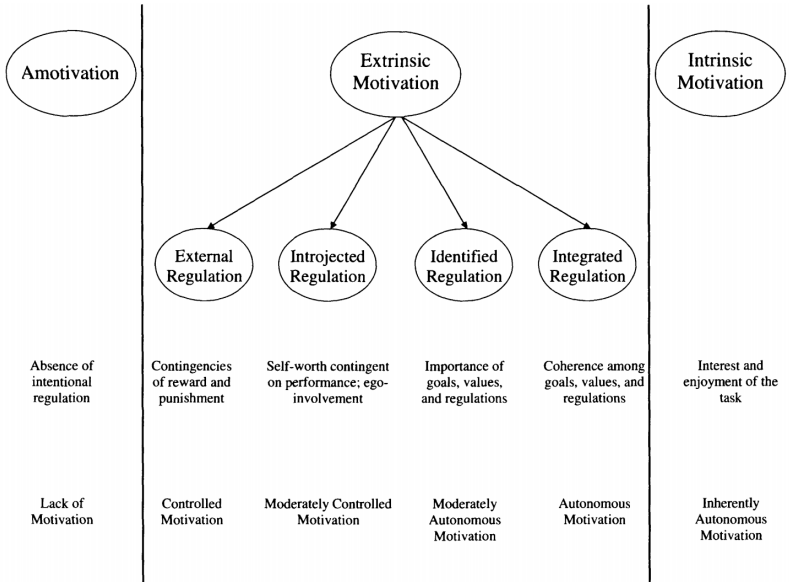


Figure 3.1: Figure of self-determination continuum according to Gangé and Deci(25).

3.4.1 SDT in Relation to Education

It has been shown that there are factors that can catalyze or undermine intrinsic motivation. Things as tangible rewards, threats, deadlines, directives, competition pressure and negative performance feedback undermines intrinsic motivation according to Cognitive Evaluation Theory. On the other hand choice and opportunities for self-direction, as well as positive performance feedback has been shown to enhance intrinsic motivation(23). Behaviours that are not intrinsically interesting to a person will require extrinsic motivation to be adopted. To make an extrinsic motivation more self determined is the process of internalization and integration. This is done when a student truly understands the values of an activity, identifies with it and incorporates it with their sense of self. This is suggested to be done by foremost addressing the basic psychological need of relatedness, by having the behaviour valued by significant others to whom they would like to feel connected. Therefore it is important to provide a safe comforting environment where the students feels that they can trust the facilitators. To further support internalization and integration it is argued that the need for competence has to be supported through challenges

where the student feel that they have the competence to succeed. To support internalization and integration to the extent that the regulation becomes autonomous however, the basic psychological need of autonomy has to be supported by the environment as well. This is suggested to be supported by the environment in such a way that makes the student feel free and agentic to explore new ideas and exercise new skills(23). It is suggested that selecting programmes, the possibility of dropping out of courses, flexible schedules and the possibility to skip classes are a few ways in which college supports autonomy in ways that high school does not. It is suggested that this might be the reason to why there are students in college that match an autonomous motivation profile, whereas high school students all fall into the category of controlled motivation profiles(26).

3.5 Computational Thinking

During the 21st century there has been an increasing interest in the field of computational thinking (CT). It started with Jeannette Wing's article in 2006 about CT and argued for this new competency in schools to enhance children's analytical ability in STEM subjects, it was not just for computer scientists anymore. This caught the attention of the academic community that started to interpret her definition and since then perform their own research on CT. Although the concept of CT being important in education is not new, as early as the 1960's there were those advocating teaching programming to college students. Most notably was Seymour Papert's MIT work with the program LOGO in the 80's, as this was aimed at K-12 education.

There are many different takes on exactly how to define CT, Wing(27) defines it as "Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent". This definition is all about how to think when posed with a problem, the abstraction and process to arrive at a solution step by step. Another definition that is more about the importance of being able to reflect on and see the modern world through the lens of CS is proposed by the Royal Society(6), "Computational thinking is the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes".

As a result of these different definitions, although highly related to each other, the following list of elements is widely accepted as containing the basis of CT in curricula that aim to asses the development and support the learning of it(28):

- Abstractions and pattern generalizations
- Systematic processing of information
- Symbol systems and representations
- Algorithmic notions of flow of control
- Structured problem decomposition

- Iterative, recursive and parallel thinking
- Conditional logic
- Efficiency and performance constraints
- Debugging and systematic error detection

It is quite evident that most of the recent work regarding CT has been about development tools and definitions, not as much focus have been made on the assessing of CT and how to do it as large gaps still exist in this part of the field(28).

The key for integrating CT in K-12 is the assessment of it. Without a method for assessing the learning of CT with students there is little hope of it being incorporated in the K-12 curriculum(28). Two models on how to do this have been acknowledged, the MIT model and the Barefoot model, accompanied with their own definition of CT as well. These models will be portrayed in further detail in coming subchapters.

3.5.1 MIT Model

Recently researchers at MIT have developed a CT framework based upon studies they made. Also by studying learners using and engaging in programming through Scratch, a definition of what CT is was split into three categories: computational concepts; computational practices; and computational perspectives(29). Computational concepts entails being able to grasp seven specific concepts that are common in many programming languages(30):

- sequence: identifying a series of steps for a task
- loops: running the same sequence multiple times
- parallelism: making things happen at the same time
- events: one thing causing another thing to happen
- conditionals: making decisions based on conditions
- operators: support for mathematical and logical expressions
- data: storing, retrieving, and updating value

It soon became clear that the concepts as a framework for CT was not enough, something to support the process of construction was needed. By studying how the learners adopted different strategies when developing their projects four distinct practices was identified. Experimenting and iterating being one, test and debug being another. Making use of existing projects or ideas and build on them was also practiced frequently and being able to see the connection of the smallest part to the whole project. The third category, perspectives, is all about the learner reaching a new level of awareness of the technology that surrounds them. Being able to express themselves and seeing computation as a medium for creation, by recognizing the power of creating with and for others and lastly be confident and ask questions about the world. In order to asses the level of CT development with the learners, as knowing the definition of a computational concept is not useful if one cannot put it to use in practice, there is three strategies that can assist(29). Artifact-based

interviews let learners engage in conversations about their projects and practices, using examples to guide the conversation forward. Another way is to provide a set of design scenarios for the learners that they engage in, giving them four different angles to relate to, critiquing; extending; debugging; and remixing. Documentation is about learners developing a sense of reflection on their own creations and ideas.

3.5.2 Barefoot Model

The Barefoot project was established in 2014 and aimed to support primary school teachers in England to get ready for the addition of CS elements in the new curriculum. Barefoot developed their own set of definitions of CT and how to apply it in a school environment. According to Barefoot(31) CT is quite simple to explain, CT is about looking at a problem in a way that lets a computer help solving it. Divided into two processes and the first being to think about steps needed to solve a problem, second letting technical skills in programming put the computer to work in solving the problem that is stated. Their interpretation of CT is composed of six different concepts, Logic; Algorithms; Decomposition; Patterns; Abstraction; Evaluation, continuing with five approaches, Tinkering; Creating; Debugging; Persevering; Collaborating, and this is similar to other models of how to think about CT.

4

Methodology

This chapter briefly describes methods used in the project. They are divided into the research part and iteration part. Research entails methods such as qualitative literature review and interviews that lay the groundwork for the project. Additionally, methods used during the design process are called the iteration methods, included are methods for ideation, prototyping and evaluation.

4.1 Research

Methods concerned with collecting both qualitative and quantitative data to diverge the design process and open new doors are here described as research methods.

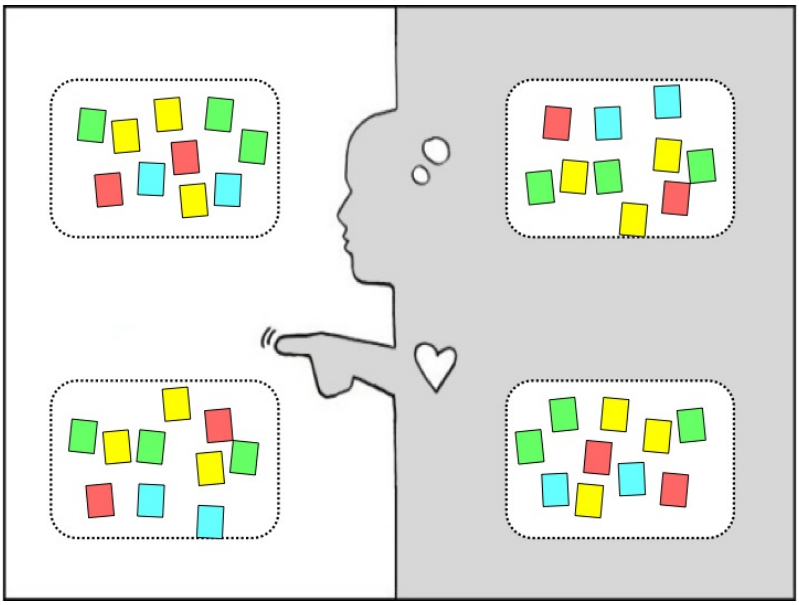
4.1.1 Qualitative Literature Review

A literature review is a vital part of the research process and needs to be done carefully. The topic of having an abundance of literature to choose from, decisions must be made on what approach to use and what to include, as it is nearly impossible to cover it all. Literature gathered can be represented in any form of the following sources: research articles, article reviews, books, websites, government documents and journals.

4.1.2 Recruiting Tools

Recruiting Tools is a method that has the designers reflect upon the way participants are being recruited to the project. It aims to ensure that a diverse set of participants are being recruited. The team ought to be aware of covering different ages, genders but on a more abstract level also differences in motivation, needs and wishes, within the group of users. Recruiting tools as a method has the designers set up a strategy for the recruiting process as a deliverable. This ought to cover strategies for reaching diversity in participants as well as legal strategies for recruiting minors under age and strategies for managing confidentiality of the participants.

4.1.3 Empathy Map



The method Empathy map is developed by Stanford(32) and aims to help understand users needs and derive insight from them. It is suggested to be performed by the researcher on a single user or a group of users. It requires about half an hour, pens and paper. Start by dividing a paper or whiteboard into four equal quadrants representing what the user: says, does, thinks and feels. The "Say"-quadrant is to be populated with user quotes and words that appear valuable and suggest deeper meaning. The "Do"-quadrant is to be populated with user actions and behaviours that have been observed. The "Think"-quadrant is to be populated with interpreted user thoughts and beliefs. The "feel"-quadrant" is to be populated with interpreted user feelings and emotions. As the last two traits are not directly observable it is suggested to infer them by paying close attention to body language, tone and choice of words.
The next step is to look at these traits and specifically look for contradicting ones, to identify human needs. That is, according to this method, physical or emotional human necessities or desires. Needs are verbs not nouns. Write these needs down on the side of the empathy map.
Next insights are to be derived. These are, according to the method, remarkable realization that help the design task at hand. Insights come from looking closely at two contradicting attributes within a quadrant or cross quadrants. Alternatively by keep asking "why" when finding strange behaviour(32).

4.1.4 Stakeholder Mapping

Stakeholder mapping is a method concerned with understanding the network of people affected by or affecting a particular system. Mapping these connections out aims at gaining a better understanding for the bigger picture of the system and see previously obscured opportunities. The method requires a diverse team and a

subject area to focus on. The team is then to generate a broad list of stakeholders, draw each one of them out as a symbol on a map, draw speech bubbles from each one of them with a short text summarizing their mindset and give them labels or titles. The next step is to draw connections as arrows between stakeholders and give these arrows a label describing their relationship. Finally circle related groupings and label them.

4.1.5 Fly on the Wall Observation

In an attempt to assess to which degree both teachers and students handle new technology and how fast they are "up and running" with it, a form of Fly-on-the-wall observation will be used. It is interesting to see how determined they are, in what form roadblocks pop up for the users and letting them handle it themselves. By taking the passive role as an observer, and not interfering with any hints, notes can be taken as to how users begin familiarizing with new technology, how they go about fetching more knowledge and how they solve problems they step upon. This can give designers helpful hints as to where the biggest issues lie when getting accustomed to a new (although similar) technology.

4.1.6 Semi-Structured Interview

Semi-structured interview is a method with perfect balance between open-ended and highly structured interviews, no set of questions have to be followed up but instead allows for the interview to diverge(33). This is a very useful format for conducting qualitative research and works equally good in the early stages of the research phase as well as the latter phase stages. Some important things to have in mind when conducting semi-structured interviews is the use of open-ended questions, nothing that allows for a "yes" or "no" answer. No leading questions and the interviewers ample use of probes to gather data at a depth are other examples of practicing semi-structured interviews(33).

4.1.7 Exit Tickets

Exit ticket is a method used to gather feedback on students understanding at the end of a workshop. At the end of a workshop the teacher prompt the students with answering a few easy questions regarding key material from the workshop, usually written down on post-it notes and handed to the teacher. This gives great value of information for the time invested by the teacher and can also work as a useful basis to guide upcoming teaching decisions, and also allows students to synthesize and integrate the information gained for their own benefit(34).

4.1.8 Personas

When it comes to creating a design that must satisfy a diverse audience of users, Cooper says "The best way to successfully accommodate a variety of users is to design for specific types of individuals with specific needs"(35). A user persona is the representation of the needs and behaviour of a set of made up users, often created

from data derived from field studies, interviews or observations. Personas can thus be powerful when distinguishing users and to highlight their needs and behaviour, which often interfere with one another, therefore identifying the right individuals to design for and making sure the most important users needs are met without compromising the needs of secondary users is crucial for a successful product. This seems very much applicable in the context of this thesis as there certainly is a plethora of different needs with the students in classes, and therefore it is vital to extract the essence of those needs and design for the wide variety focusing on the important needs. The usage of personas have been greatly increased in the user experience community, since it can act as a multipurpose design tool and aid designers in problem solving.

4.1.9 Journey Map

Journey mapping is a tool based on the premises that humans are more able to relate to narrative rather than pure data. Hence personas are used to describe different user groups and the journey map acts as a narrative for these personas to travel through. This can be used to communicate an existing user experience, as well as describe an envisioned future optimal experience. It can aid in weighing the impact every elements of an experience, e.g interactions, decisions or emotions. It is also a good way to display your understanding of different situations(36). Journey maps are commonly illustrated as different personas trajectories through emotional states and expectations, as well as their touch points with the organisation. A journey map spans over a certain time section, it can be the user's entire experience with an organisation from first contact to end, or simply a single section in time. A journey map is used to identify points in time where users interact with the organisation, so called touch points.

4.1.10 Affinity Clustering

As described by the the LUMA Handbook of Human-Centered Design Methods(36) affinity clustering is "a graphic technique for sorting items according to similarity". Starting out with a set of data the team is supposed to write individual items on post-it notes. Next the team should read each item out loud and attach it to a wall or table where it's relation to other objects can be discussed. Notes argued to be related are supposed to be grouped with proximity. Finally the groups are to be labeled according to content, as this allows for new abstracted patterns to naturally emerge out of the data set. For this method it is also advised to have a diverse team.

4.2 Iteration

Methods for iteration describes methods that are used both to ideate, make prototypes and evaluate the results.

4.2.1 Brainstorming

After the research phase the design team needs to begin ideating over possible design solutions. The potentially most used activity is some form of brainstorming, where the participants get together and spit out ideas in a frequent, no filter and fast paced way. In order to have a fruitful brainstorming session a certain mindset must be instilled in the participants, along with some rules to follow. There must be no boundaries on idea solutions, the mind must be allowed to roam freely. There should not be any emphasis or thought of seeing the ideas of feasible solutions, that is not what brainstorming is about. If a brainstorming session is successful the team should have a lot of ideas to put a spin on afterwards, and with further work have them contribute to a feasible design solution. Also some preparations regarding what the team learned from the previous phase, the research phase, is necessary. This can be done by finding themes in the groundwork that has been done already through interviews and field studies.

4.2.2 Design Principles

Insights gathered and design themes identified can be put to good use for the remainder of the project, by turning them into design principles. By having design principles the design team always has something to fall back onto, and can feel confident that as long as the designs produced stay true to the principles, the designs will be relevant. It is important to try and keep the principles short and memorable, e.g “Talk like people talk” or “Keep women at the center of business”.

4.2.3 Integrate Feedback and Iterate

What was learned about the users in the observation phase can be further investigated in the iteration phase by showing them the prototype and finding out what they think. By integrating their feedback into the design work and developing another prototype is a great way of refining the idea into something that is a finished product. Integrating feedback and iterating is closely tied to rapid prototyping and it is therefore important to start building on the next prototype as soon as the designer is settled on what should change, drawing from reflections on the feedback received. It is mainly a method for refining the idea and is bound to be used several times over as the results will increase the chances of the right solution being close.

4.2.4 Abstraction Laddering

A useful method when trying to identify the levels of abstraction in a problem is abstraction laddering. This method helps designers concretize how to solve abstract problems, and turn concrete problems more abstract by asking why and is a great tool for comparing solutions and evaluate them.

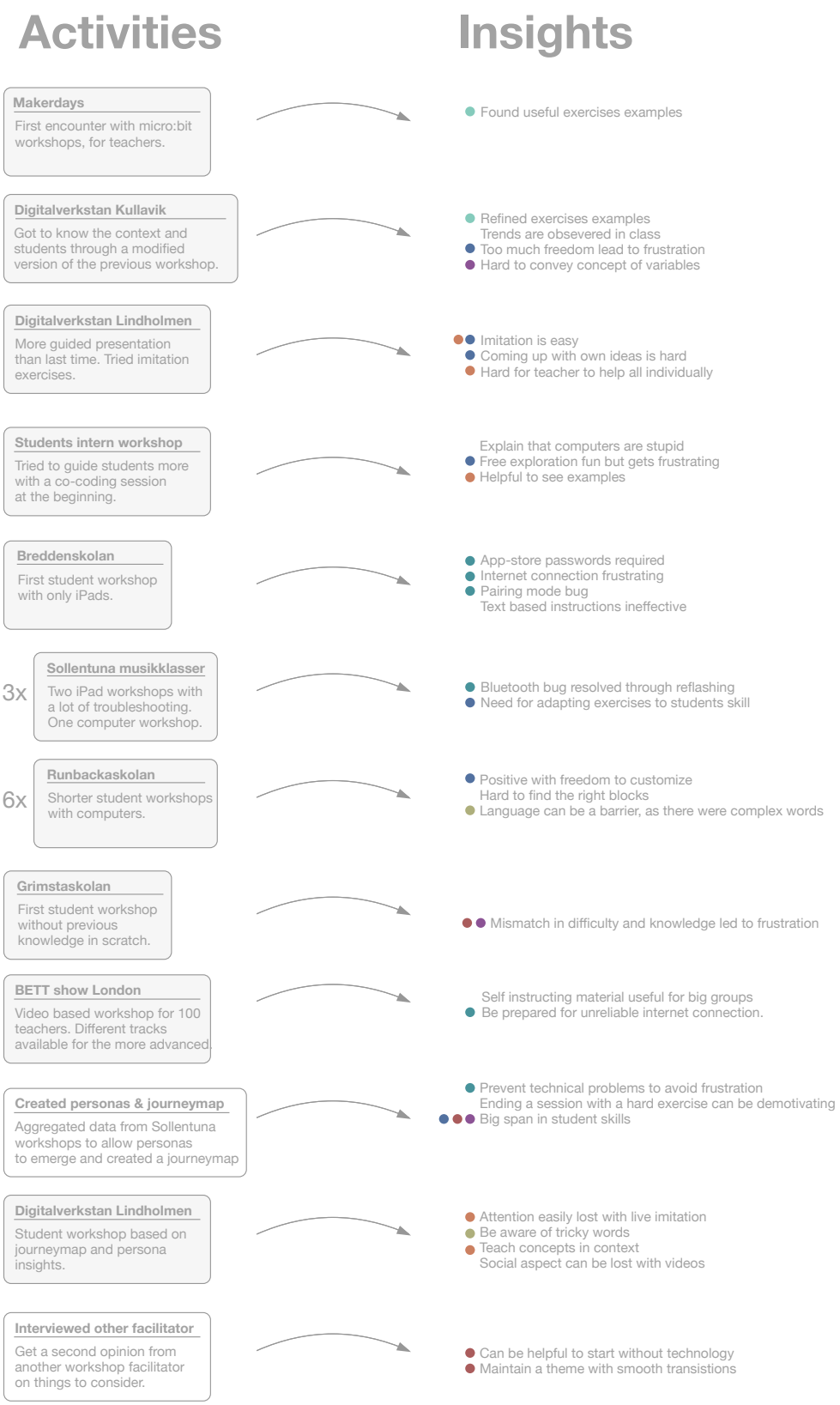
4.2.5 Rapid Prototyping

Rapid prototyping is an excellent tool to quickly learn through making and turning ideas tangible as you get feedback from targeted users. This method is only meant to roughly explore an idea, not be perfect, so you do enough work to test the idea and make room for improvements next iteration after gathering feedback. This method both builds on previous knowledge gathered in other methods but also allows for tinkering with lessons learned from recent prototype testing.

5

Process

This chapter describes the measures undertaken during the design process, practicing various methods and continuously reflecting on previous results to feed into subsequent activities, in an attempt to eventually find an answer to our proposed research question. In *planning* we describe how the thesis plan unfolded and the decisions made for choice of methodological framework and time frame, additionally our thoughts on the literature and theories we indulged in. All our activities are documented in detail, the origin of each workshop, purpose, planning,result, data gathering and possible insights are also presented. For the first half of the thesis we had a different approach to workshops than later on in the process, due to the need of data gathering and getting to know the users. We discuss some methods we used to extract insights from the data gathering and move on to plan a set of workshops where we followed the same class over a couple of weeks. During the insight analysis we describe methods used to finally condense all our data into something tangible and how this was reiterated after getting feedback.


Activities
Insights
Makerdays
First encounter with micro:bit workshops, for teachers.
Found useful exercises examples
Digitalverkstan Kullavik
Got to know the context and students through a modified version of the previous workshop.
Refined exercises examples
Trends are obsevered in class
Too much freedom lead to frustration
Hard to convey concept of variables
Digitalverkstan Lindholmen
More guided presentation than last time. Tried imitation exercises.
Imitation is easy
Coming up with own ideas is hard
Hard for teacher to help all individually
Students intern workshop
Tried to guide students more with a co-coding session at the beginning.
Explain that computers are stupid
Free exploration fun but gets frustrating
Helpful to see examples
Breddenskolan
First student workshop with only iPads.
App-store passwords required
Internet connection frustrating
Pairing mode bug
Text based instructions ineffective
3x
Sollentuna musikklasser
Two iPad workshops with a lot of troubleshooting. One computer workshop.
Bluetooth bug resolved through reflashing
Need for adapting exercises to students skill
6x
Runbackaskolan
Shorter student workshops with computers.
Positive with freedom to customize
Hard to find the right blocks
Language can be a barrier, as there were complex words
Grimstaskolan
First student workshop without previous knowledge in scratch.
Mismatch in difficulty and knowledge led to frustration
BETT show London
Video based workshop for 100 teachers. Different tracks available for the more advanced.
Self instructing material useful for big groups
Be prepared for unreliable internet connection.
Created personas & journeymap
Aggregated data from Sollentuna workshops to allow personas to emerge and created a journeymap
Prevent technical problems to avoid frustration
Ending a session with a hard exercise can be demotivating
Big span in student skills
Digitalverkstan Lindholmen
Student workshop based on journeymap and persona insights.
Attention easily lost with live imitation
Be aware of tricky words
Teach concepts in context
Social aspect can be lost with videos
Interviewed other facilitator
Get a second opinion from another workshop facilitator on things to consider.
Can be helpful to start without technology
Maintain a theme with smooth transistions

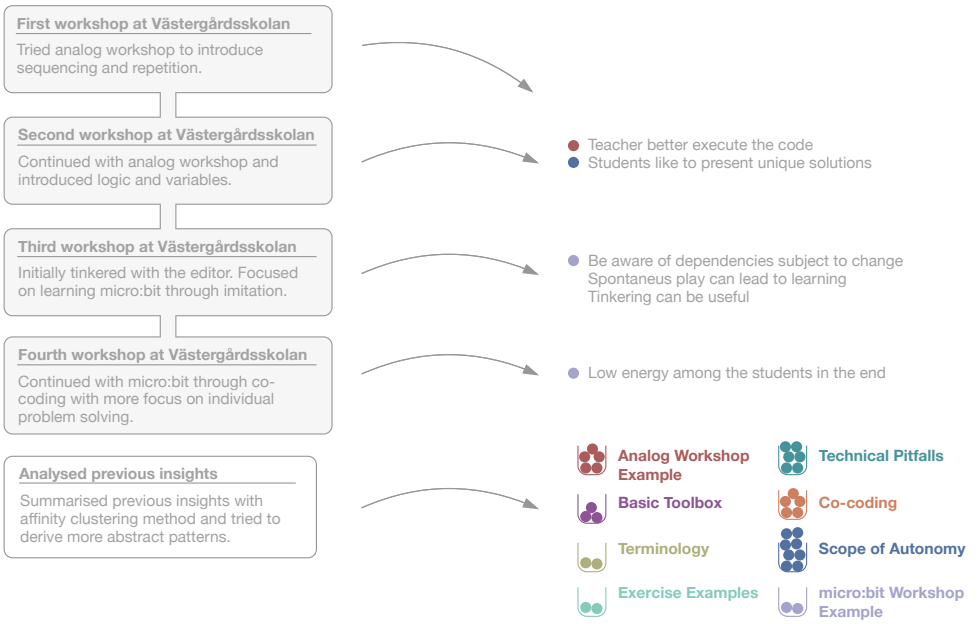


Figure 5.1: A model showing performed activities and the flow of insights throughout the project

5.1 Planning and Pre-study

In this section we describe how the planning of the thesis unfolded and how we chose methodological framework and time frame, furthermore our thoughts on the literature and theories that we indulged in. Additionally we mention our first workshop, where we got to know the nature of facilitating a workshop and handle an audience.

5.1.1 Planning

Having determined our first version of the research question we needed to gather fundamental knowledge, both theoretical and practical, that concern micro:bit and using technology to aid education. In addition we investigated different options for a methodological approach as well, previous experience with the human centered process and the prospect of the thesis being of an iterative nature, HCD felt like a solid choice. Although small adjustments to fit our idea and plan was made. We divided the design process into three separate phases, research, iterative and demonstration.

The research phase was all about acquiring the right information and understanding of the work that was to be carried out. That entailed reading about earlier work in the field and learning about theories connected to teaching. Since both teachers and students were the focus, information gathering about how they perceive the introduction of programming in school was of high priority. Additionally we also familiarized us with the platform, micro:bit, during this phase. After the research was done we continued to our iterative phase which consisted of three sub-phases,

ideation; intervention; and evaluation. These sub- phases were one prototype cycle, and the aim was to refine the design every time a full cycle was completed. In the beginning we aimed for only a couple iterations, but we ended up exceeding that. We had a good frame for iterating and gathering data so we used it with ease almost every time we had a workshop planned.

Ideation phase is where previous knowledge came to use when designing for the objective at hand. During this part we created plans for upcoming workshops and activities. This was made through a script that we used, consistently together with all workshops that we did. We also used insights from evaluated data to create new content. At a later ideation stage we created personas from data gathered and used them in a journey map in an attempt to extract even more insights. These methods were not included in the plan from the beginning.

In the intervention phase the implementation of the design work for the current objectives was carried out. Implementation mainly comprised of field tests, namely workshops in school environment on several occasions. While field testing the prototype feedback from the testers and the process was collected for later evaluation. During the workshops most of our data gathering took place, almost exclusively through exit tickets that were to be evaluated at the next stage. Evaluation was carried out at the end of each prototyping cycle to assess in what extent the implementation was done as design implied. Also evaluating feedback and other input from testing the implementation in a live setting, if that was the case. This is also when we evaluated the data gathered, and tried to extract insights we could use to prepare for the next cycle.

In our final evaluation phase, when there would not be any more iterations, we began evaluating all our data with affinity clustering in an attempt to reach a theoretical result, which was something not included in the plan from the start but emerged as a necessary method to evaluate the data. This itself was done over a few iterations and led us in the end to a model of autonomy. We also found insights and valuable information for teachers that did not fit into the model, and we made them into a bundle of considerations when working with micro:bit. Last phase of the master thesis planning was about demonstrating the results. This step highlighted what important elements to consider when designing digital teaching material regarding the micro:bit suited for a primary school classroom in Sweden, as an answer to the research question. The suggestions produced were based on work done in both research- and the iterative prototype phases. This phase also entailed the finalization of the project report and presentation.

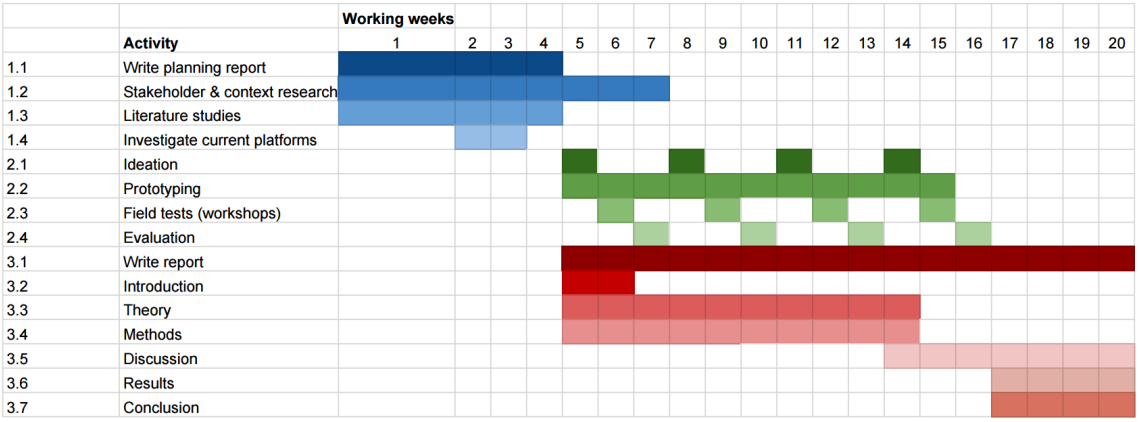


Figure 5.2: Planning diagram with weeks and acitvities

5.1.2 Literature Study

There were several aspects of learning and teaching that we needed to gain knowledge in. Since we had no previous knowledge in the underlying theories that regard teaching and learning we needed to do some groundwork. We began by creating a research document where we put down topics that could be of interest and theories connected to them. We asked ourselves if it could benefit us in pursuit of answering the research question, so we also did some culling of the topic ideas. The topics we finally came up with were digital literacy, teaching pedagogies, maker culture, digital fabrication, and micro:bit. Since the thesis is based on the introduction of digitalization in Swedish primary school education and preparing the younger generations for the future by making them digitally literate, we felt that digital literacy (or computational fluency) was of high interest. On top of that we needed to learn how they already are teaching programming in an effective way to younger students, what approach one should have and where to put focus. This led us into the field of computational thinking and teaching pedagogies. Theories about computational thinking gave us guidance on how to approach teaching of programming for beginners and sorting out the different programming concepts. Regarding teaching pedagogies, there is an extensive amount of research in that area. During the thesis we came across several interesting phenomena during workshops but had to limit ourselves to the above mentioned theories, since allowing for another school of thought would have rendered too much extra time investment. We chose to explore theories on self determination in education and constructionism as they felt closer to maker culture and digital fabrication, which is something that we associated to introduction of programming and working with micro:bit. As there is a plethora of theories touching teaching and learning, we had great use of a meta synthesis of over 400 learning strategies(22). Studies that showed that the classroom was changing with the digitalization, teachers might have a different role to play and the maker culture phenomena(3), got our attention as well.

5.1.3 Makerdays

RISE Interactive, our business client, arranged a conference called Makerdays which we were encouraged to help out with so we could learn more about facilitating workshops and get to know teachers first hand. Makerdays is a meetingplace for teaching and exploring in the borderlands of making, digitalization and creativity. Makerdays is aimed at those involved in Swedish schools, science centers and other pedagogical activities. The event lasted for two days and hosted about 300 teachers in total, of which a third had chosen to try the micro:bit. This resulted in 5 workshops, of approximately one hour each. These days were used as an open ended early exploration phase to learn more about hosting workshops and to better get to know teachers.

Prior to the conference we had a few days where we tried to come up with interesting exercises, projects and challenges that could be done with the micro:bit. Even though we did not know it at the time, these would become the exercises that we would use later on throughout the project.

During the workshops we assisted our colleague, who was the head facilitator of the micro:bit workshops. The workshops began with a presentation of the micro:bit and its features, we also showed a couple of projects that we had prepared to give the audience ideas of what is possible to do. We ended the presentation by coding a name badge on the micro:bit, a simple program where the micro:bit scrolls a string of text on the display, then we let the teachers have the rest of the time testing it out themselves. A set of challenges were provided to trigger their imagination.

5.2 First Sessions with Digitalverkstan

Through previous collaboration outside of the thesis work we got in contact with Digitalverkstan, together we were able to arrange two workshops and introduce micro:bit to primary school students. Both workshops took place during the same week but at two different locations, Kullavik in Kungsbacka and at Lindholmen. The plan were to get our first hands-on experience in facilitating a micro:bit workshop with our users.

5.2.1 Workshop at Kullavik

Digitalverkstan Kullavik

Description:	Explorative workshop with students of Digitalverkstan
Time:	25th October 2016, 18:00-20:00
Location:	Kullaviks Montessoriskola, Kullavik
Participants:	Group of around 30 children, 2 facilitators, 1 teacher and a few parents helping out
Age:	8-10 years old
Aims:	- Get hands on experience of the role of the workshop facilitator - Explore the actual context and meet the real users - Gain understanding for the current level of computational knowledge of the users
Methods:	Execute planned workshop with presentation, discuss and note down observations after

Insights:
- Refined exercise examples
  Trends are obsevered in class
- Too much freedom lead to frustration
- Hard to convey concept of variables

Figure 5.3: Script snippet of the workshop

Our first workshop with Digitalverkstan was done in Kullavik just south of Gothenburg, with about thirty students between eight and twelve years old. We went there to get hands-on experience in being a workshop facilitator and explore the actual context as well as meet the users. We had made a plan which was loosely based on previous workshops during Makerdays, since we had no previous experience to relate to we used it as our base. Therefore we began with a lengthy and detailed introduction via a presentation, showing the ins and outs of the micro:bit. We ended the presentation by showing how to code and upload a simple program, a name badge. When done micro:bits were handed out and they were given some time to program their own name badge onto a micro:bit. After a while we also showed the students how to use inputs on the micro:bit, e.g buttons, and asked them to use that in their name badge program. From there it was more or less a free exploring workshop, although we provided a list of programs they could try and complete, and we went around and provided help for those who had questions or were stuck. Beforehand we had done research on computational thinking and therefore wanted to gain an understanding on what level the students were at. We had a plan to use a rubric to gather that data, but there were simply no time. We had underestimated how much time helping students would consume, and at one point during the workshop we decided to prioritize the students and help them learn the micro:bit.

In hindsight we realize we might had progressed too fast, the students were not ready and got stuck and confused. As a possible result of that we observed when a pair of students managed to complete a cool game, this time rock-paper-scissor, the other students looked to them and eagerly wanted to do the same. Soon everyone were trying to code their own so they could play against each other. This also led to many students tended to focus on the result rather than the process of learning, they basically wanted us to make the game for them so they could play with their friends. This kind of behaviour has both positive and negative impacts, the good part being that they get motivated and suddenly have a purpose to learn how to code a micro:bit. On the other hand they often get impatient and try to skip a few steps in the learning process to reach the end result faster, which is not always desired from a teacher perspective.

5.2.2 Workshop at Lindholmen

Figure 5.4: Script snippet of the workshop

Coming into the second workshop we had adjusted the workshop with recent findings in mind. The conditions were similar to the last workshop, with primary school students around eleven years old and new to micro:bit, only the location was different. This time we wanted to emphasize a few concepts related to programming, e.g variables, in order to give them more tools when programming by themselves later. Therefore the introduction part was extended. First we went through by demonstrating the name badge and explained the blocks used in more detail as well as how to upload code to the micro:bit. We continued by adding buttons to the name badge, talking about input blocks and how they can be used. To include randomness and variables in the introduction, example programs of coin flip, counters and dice were shown and described. Extra care was put into explaining variables, as the concept is abstract, and we used the common box analogy for that single purpose. One popular way to think about a variable is to imagine a variable is like a box that can hold values, with the box label being the name of the variable.

When we were done with the introduction we let them play with their own micro:bit, although this time they had seen how to modify and manipulate blocks and code. First they were asked to make their own name badge and add some feature from the examples give, e.g a button press. All examples that we programmed during the introduction were also provided as inspiration on a slide, if they were stuck they could copy the code. On top on readymade examples being shown, a set of challenges were given as in the last workshop. These had no slides showing the code but all the blocks and knowledge they needed to solve had been shown in the earlier examples, they just needed to figure out how to combine them. This method had us distinguish two kinds of students in our workshops. Some students were able to follow during the introduction and later complete the exercises, but were bewildered once they tried to solve one of the challenges. Others had no problem combining blocks from the different programs to solve a challenge. There was an apparent risk of students just copying the programs mindlessly without paying attention to learning, and we addressed this in our evaluation of the workshop. To increase the amount of students that understand what they are coding we thought of scrambling the blocks needed, still giving them the right blocks but unassembled.

5.3 Workshops with Student Interns

Figure 5.5: Script snippet of the workshop

An opportunity arose at our client, RISE Interactive, where they were to have two interns in their office for a week. Although these students were 9th graders and a bit older than our target group, which is 4-6th graders, we felt that this still was something that could help our research progress as they had no previous experience in programming. We used this occasion to test out some ideas we had regarding our micro:bit workshops in general and application of new theory we had acquired about learning strategies(22). First an activity plan and a script was created for the workshop, containing our own goals as well as details on the activities it should hold. The workshop began with an introduction to micro:bit and us showing how the editor works, we also presented a success criteria for them to work towards during these two-day workshops. The success criteria were the following:

- They come up with their own idea for something they want to make
- They pursue the knowledge needed to implement their idea, in a gritty independent fashion, where they take responsibility for their own learning
- They use the knowledge to create their ideas
- Their creations help them to generate new ideas

Next a co-coding session was planned, where we solved problems and coded together with the interns to ease them into programming with the micro:bit. We coded programs that we had used before with beginners, e.g name badge and dice, to teach the basic logic in programming. Dice is another simple program you can do with the micro:bit to resemble a six sided dice, it includes using variable, input and sequence blocks. Co-coding was a positive experience for both parts, the interns got their questions answered and were able to follow the steps needed to solve the problem with less pressure on them, and we got to test their skills in a relaxed fashion. When we had finished the basic knowledge teaching we let them work on their own, with their own project. They did this for the rest of the day and continued for a few hours the next day, then we gathered feedback from them through a semi-structured interview. As a result we realized that we had given them too much freedom after the introduction and co-coding. They felt they did not have enough

to come up with an idea on their own, this lead to a suggestion of a middle step between co-coding and working by themselves. By creating problems with already set blocks, although scrambled, we might bridge the gap from being comfortably supervised to having to solve problems all on your own. They also admitted that it is easy to learn by looking at examples, but the step from imitation to a blank canvas is too steep and therefore a middle step or a gradually increasing freedom of work is the key. As they both were new to programming they also felt that anything that works, no matter how trivial, is a good way to start in order to build confidence so you may dare to try and fail later on.

5.4 Workshops in Stockholm

For our research we needed to gather relatively large amounts of data on the needs of students and teachers regarding programming in school, and especially the micro:bit. We got in contact with a set of four schools in Sollentuna, north of Stockholm, that were interested in participating in our study that also included teacher workshops. Teacher workshops had been overlooked since Makerdays and we felt we needed more data in that area, so these workshops served as a great, and much needed, opportunity for us. The plan was to visit one school each day from monday until thursday during one week and host a total of twelve workshops, to twelve different classes spanning from 4th to 6th graders. Our workshops varied in both length, content and approach since classes were different ages, some had previous knowledge in programming and some not, the workshops duration varied from 45 minutes up to almost two hours. Half of the schools used iPads in their teaching so we had to adjust our material to that as well. As data gathering methods we used exit tickets after each workshop with the students and also did a short video reflection with our own thoughts. With the teachers we used an empathy map to gather their thoughts and ideas.

5.4.1 Preparation

In order to prepare for the workshops in Stockholm we began constructing a plan for our visit. We built a timeline where all the classes and workshops were presented so we easily could get an overview, see figure 5.6. On this timeline we placed different constraints, such as time schedules, age groups and platform to work on (iPad or PC). Then we began thinking of the content to provide, which eventually led us to a model of using different workshop modules as entities to move around to tailor each workshop with. It was not certain in the beginning if we wanted to do each workshop different and compare those results or make them similar and look at differences. We placed these modules, in the form of different colored post-its (one for each module), on a large paper to arrange the workshops. The modules we decided on were Introduction; Co-Coding; Exploration; and Evaluation. Introduction was comprised of us going over the basics with the micro:bit, showing students how to connect and upload code and showing simple examples like the name badge. This part was extended with classes that used iPads, as connecting with bluetooth is less

trivial and demands more guidance than using usb. We even prepared and printed instructions on how to pair the micro:bit via bluetooth.

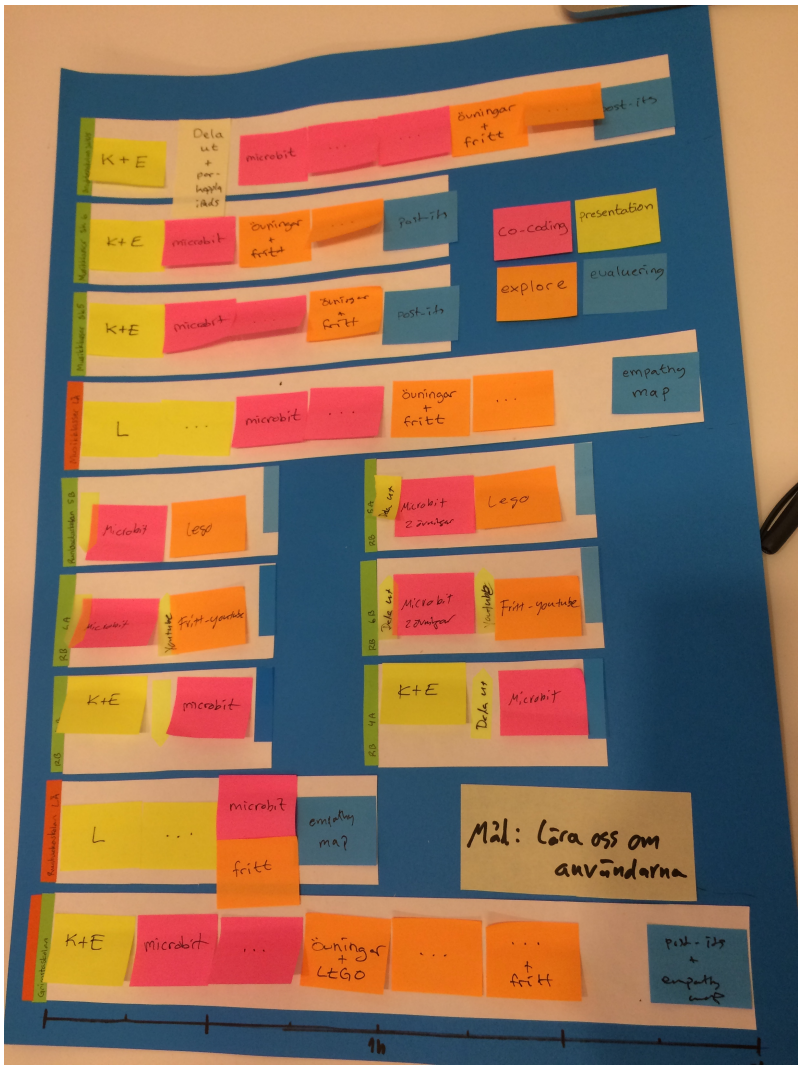


Figure 5.6: Preparation timeline for all workshops

Next we let the students code the micro:bit together with us while co-coding, we began by letting the students make their own name badge program and make sure all knew how to upload code to the micro:bit by themselves. After that we stepped up the difficulty level a little bit to be able to show off more features of the micro:bit. We had a set of readymade programs with rising difficulty that we followed, which were programs that we had used in previous workshops, depending on time available and students knowledge the amount of examples we presented varied from class to class. Exploration was about the students being able to program the micro:bit on their own, most often following some kind of instructions that we provided up front or just coming up with their own idea. Instructions were in the form of challenges, most often scrambled blocks that students put together to make a complete program. All the challenges we provided were built on knowledge we had shown through the

co-coding, e.g how variables or input works.

Lastly we needed some way to gather data, which resulted in the Evaluation module at the end of each workshop. We chose to use exit tickets with the students as they do not take much time to answer and is also easy to evaluate. For teachers we felt we could get more value out of an empathy map, as they could sit in groups and discuss as a larger part of the workshop.

5.4.2 Breddenskolan

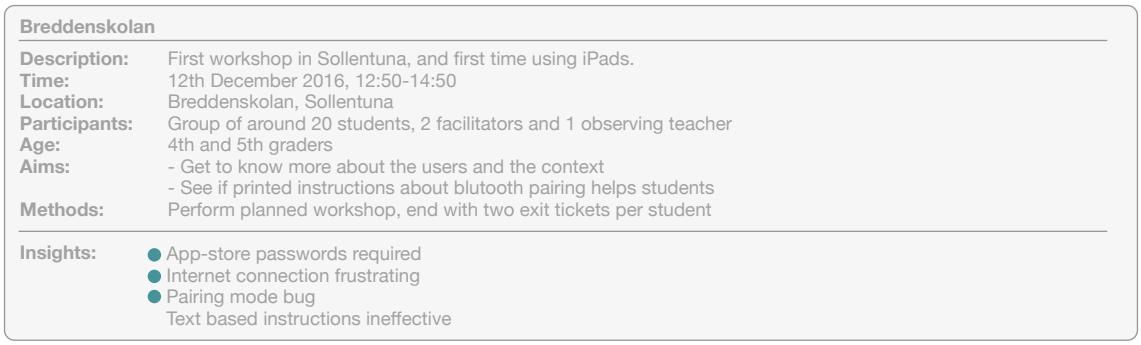


Figure 5.7: Script snippet of the workshop

The first class we visited on our trip to Stockholm was 4-5th graders at Breddenskolan with around twenty students. We knew they used iPads as their platform so we had adjusted our material beforehand, although we forgot about the fact that they needed to download the micro:bit app from appstore in order to pair their micro:bits. This led to some time wasting as they needed individual passwords for each iPad passed around from the teacher as the students normally is not trusted with those passwords. We had them sit in pairs, two students per iPad and micro:bit, before we began our introduction. We used step wise instructions on how to pair the micro:bit with the iPad, having the students follow and confirm that they were following each step. This was a very slow and tedious way of showing how to the bluetooth pairing work, partly because we needed everyone to be on the same step at the same time and we experienced some technical issues which had several students waiting for everyone to catch up.

We decided afterwards that we should show a complete walk-through of the pairing process up front next time. The printed instructions on the pairing process was barely used by the students, perhaps the design contained to much text for it to be accessible, and they rather asked for our help if they got stuck. Other than a few mishaps and unstable internet connection the atmosphere was very positive and they seemed to have fun during the workshop.

5.4.3 Sollentuna Musikklasser

Sollentuna Musikklasser

Description:	Series of 2 iPad workshops with students and 1 workshop for teachers
Times:	13th December 2016, 9:40-11:00, 11:45-13:15, 14:20-16:30
Location:	Sollentuna Musikklasser, Sollentuna
Participants:	Around 30 students, 2 facilitators and 1 observing teacher. Teacher workshops had around 15 participants.
Age:	4th and 5th grade students
Aims:	- Get to know more about the users and the context
Methods:	Perform planned workshop, finish with exit tickets for students and empathy maps for teachers

Insights:
- Bluetooth bug resolved through reflashing
- Need for adapting exercises to students skill

Figure 5.8: Script snippet of the workshop

Next school we visited was Sollentuna musikklasser and three workshops in total were planned with 5-6th graders, including one with only teachers present. These classes also used iPads as their platform except for the teachers. From the day before we decided to cut the slides describing the pairing and delayed handing out micro:bits until we had done the pairing walk-through thoroughly, ensuring their full attention. We also shortened the co-coding part since we knew the students had some experience in programming with Scratch but also because it was too long to begin with. When co-coding a bigger emphasis was put on having a dialogue with the class instead of performing a semi presentation. This got the students more involved and created a relaxed atmosphere in the classroom where you were allowed to try and test.

Shortening the co-coding part meant that the students had more time during the exploration phase, doing exercises and challenges in pairs. As a result the gap in progress between the students widened, and while a majority still were busy completing the challenges we had provided and asking for help, many had run out of exercises to do. Even though these students that finished all of the exercises were in minority, we realized that we need to account for this in the future.

At the end of the day we had a workshop with only teachers participating. Some changes were made to the content in the presentation to better suit their interests, e.g sections about the teacher's role in micro:bit workshops and correlations to the curriculum. There were about fifteen teachers present, each with their own computer. We proceeded as usual and ended with explorative phase and an extended evaluation phase. The workshop itself did not generate any exciting insights and was rather a means to get the data from the evaluation, through an empathy map. The participants were divided into three groups of five and were given a large sheet of paper with an empathy map and around twenty minutes to complete it. Their task was to discuss questions on what they think about the whole programming in school topic in general, and their role as a facilitator and the one to educate within this subject in the near future. Their interest in this topic was mixed and we felt that we might not have gotten through to them fully.

5.4.4 Runbackaskolan

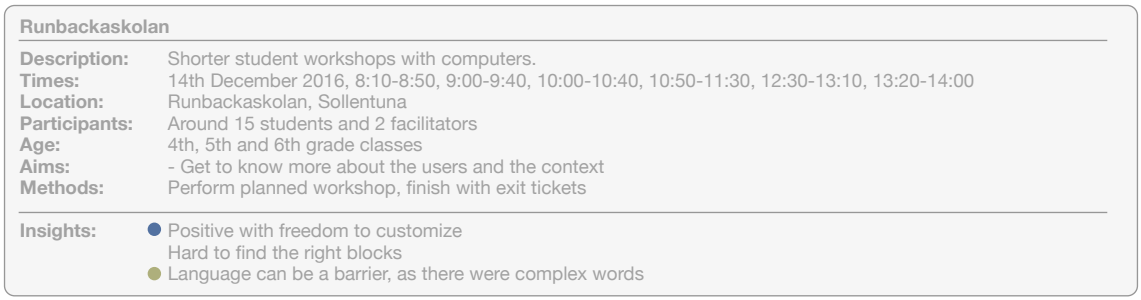


Figure 5.9: Script snippet of the workshop

At Runbackaskolan we had different conditions, only 40 minutes were designated for each workshop. There were six workshops total, two for each grade (4-6th), and we did two workshops back to back before a longer break. That gave us some time to make adjustments to the content if we wanted to. Thankfully they all used chromebooks as their platform, which saved us some time when it came to show how to upload code. First up were the 5th graders. As planned out beforehand we had introduction and showed them how to connect the micro:bit, made a simple name badge and uploaded the code. We then had an extensive co-coding session where we coded a few programs together with the class, they followed and imitated on their own computer. At the end they were given ten minutes to explore freely, although we provided scrambled rock-paper-scissor code that most of them tended to try out during that time. We planned it this way since we wanted to give them a proper introduction on the micro:bit while also follow their learning closely, through co-coding, as the timeframe for each workshop was shorter than usual.

After the first two workshops we felt that we might have underestimated their capabilities in picking up knowledge and learn the micro:bit. Because of this some changes were made to the set of workshops, we wanted them to have more time to work freely and explore and made room for this by cutting out some of the co-coding parts we felt were superfluous. For the next two workshops with the 6th graders, we gave them more time to program freely at the end allowing for more students completing the rock-paper-scissor game. This led to a fun and positive atmosphere ending the workshop with many students expressing that they wanted to extend the workshop and continue coding. We also realized that with students that have experience from block programming, e.g Scratch, we could maintain a fairly high tempo in our co-coding and they would be able to keep and continue to code by themselves in the end. The last two classes had the youngest students. Since the last workshops went so great, with slightly higher tempo and shortened co-coding, we figured that we rather keep it and see if students will receive it differently. It resulted in us having to provide more help during their free exploration, but overall a very positive experience nevertheless.

5.4.5 Grimstaskolan

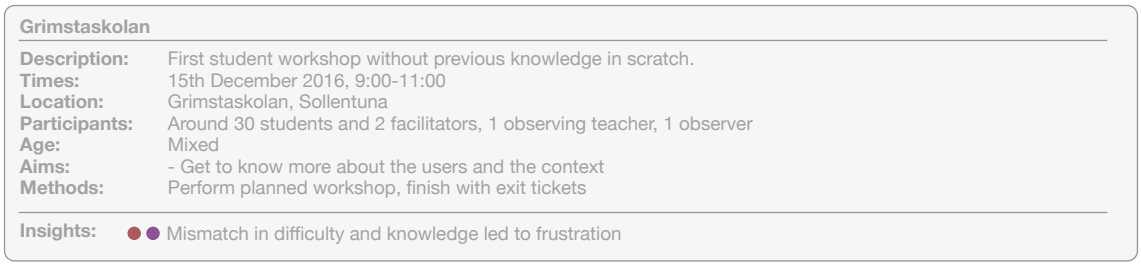


Figure 5.10: Script snippet of the workshop

As our last school we visited Grimstakolan and their 7th graders. Even though they were completely new to programming, no previous experience with Scratch or similar platforms, we proceeded with the same workshop plan. The introduction and co-coding phases went relatively well, and since we had up to two hours available for this workshop we could add content to the co-coding to make it richer. When we let them code for themselves we discovered that the plan was not well suited for complete beginners, even if they were older than previous classes. They just did not have much of a clue of what to do next. We realized that we should have done better groundwork, perhaps giving them a purpose of programming and putting it in a relevant context for them. Instead they were stuck with ideas far above their competence, since they had no clue what is possible, or were starved with ideas beyond the examples we provided for the same reason. The workshops went on for about two hours, without any break, and in the end when we tried to wrap up and show solutions of some of the problems together with an explanation, we could see that the energy was really low. We did finish explaining the code and tried to get answers or a discussion going but got very little response.

5.5 BETT Show in London

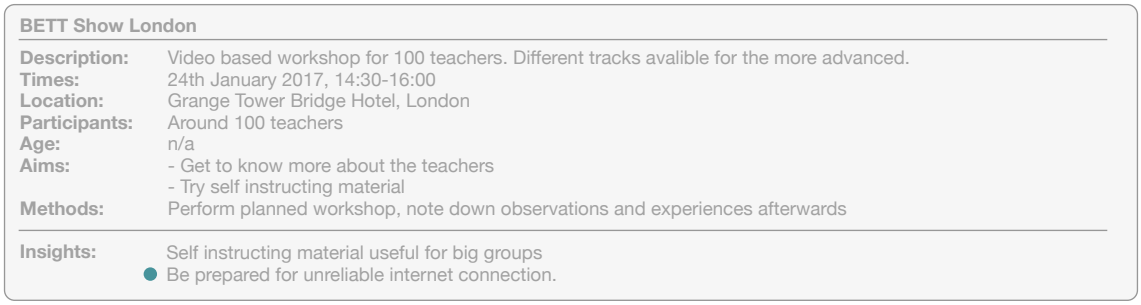


Figure 5.11: Script snippet of the workshop

During our visit to the BETT fair in London we were able to hold an introductory workshop with teachers. Since the amount of participants was around 100, and

we were only five facilitators, we had to make some adjustments to make it work. First of all we thought that in order to facilitate a functioning workshop for such a large group we would need to have some sort of self-instructing material, since we will be unable to provide sufficient help otherwise. Not all teachers present practice in the 4-6th grade classes so we had to accommodate for their interest as well. This led to us creating material for three workshop tracks, block programming (lower grades), text based programming (grades above 6th class), and a workshop for experimenting for the curious and already initiated participants. We used video material from Makermovies.se (37), made by our colleagues, and accompanied the videos with add ons to the exercises and challenges. These tracks were available through a tinyurl that we distributed during the workshop and routed the users to a presentation, once they were there they chose which track to follow. There were no restrictions in which platform to use, and we saw many experimenting with their smartphones, iPads and computers. All in all it was a very positive atmosphere and experience, although the internet connection on the premises was poor, which led to some frustration. Since we had only prepared one kind of workshop material, we realized that it would have been necessary with a back up plan for those unable to take part of the video slides.

5.6 Persona Creation

We wanted to map students behaviour in certain situations and therefore decided to create a journey map. We wanted to build our journey map based on different personas to cover more than just the average students needs.We used all our exit tickets from the previous workshops in Stockholm as our data, and began sorting them using the affinity clustering method. When we were done four different personas had emerged.

First we had a persona called Lisa. She acts as her friends are and does what is expected of her from the teacher without any real objections. She also works hard when given clear directives. One of her fears is falling behind her friends and not understanding what they know, she also lacks a natural way of doing self reflection. We also identified a more free thinking version of Lisa, which is Nina. She is more aware of her skills and does not get embarrassed for her shortcomings. She is more independent and has more grit than Lisa, likes to stay positive and prefers to experiment and try new things before asking for help. As one of the extremes we found David. He already has knowledge in the subject and is very keen to show it. As he constantly needs new challenges there is a need to allow assignments to be tweaked and built upon to not loose David's interest and spark. As he likes to show his skills, the teacher can ask for his help to aid fellow students with their assignments. On the other extreme we have Olle. A student that does not care for the subject, perhaps because there is no interest to begin with but most often the reason is lost confidence. Olle rather does nothing at all than feel stupid for not being able to solve a problem, he may have tried but the threshold was too high so he gave up. We also identified technical issues leading to frustration with the students. This made them lose interest, impatient and unfocused. This could be

induced by the internet connection being slow, bluetooth pairing or uploading of code taking too much time, or simply being lost and not knowing what the problem is.

5.7 Journey Map

A decision to create a journey map was an attempt to condense and summarize insights gathered throughout the research phase, and in turn create representations of students current experience. The ambition being that it would help inspire the creation of envisioned optimal experiences at a later stage.

Figure 5.12: Journey map with personas on the left

We used the four personas we had created and made up a scenario they could be applied to. The scenario consisted of ten different events that took place during a first workshop with micro:bit. We placed the one persona on each row, and made one column for every event. Then we proceeded to interpret the person's attitudes

towards each event, and started to populate the matrix with blue post its. When we felt satisfied with the result we began scanning the matrix for insights, which would be identified as the students needs for a special situation.

In the event of a first exercise with micro:bit we found several needs for our personas. Olle needs a slow and easy start, to be able to gain confidence and believing that this might be possible and fun for him to engage in. Lisa as well as Olle needs clear guidance to get going and not lose interest, simple and short exercises so she can feel that she is making progress fast. On the other end David might feel the introductory part to be too trivial and may risk losing interest unless he is stimulated with exercise add ons or challenges. When posed with a first problem to solve, Nina needs time and space to explore possibilities for a trial and error approach. She does not wish to be presented with the answer when asking for help, but rather something to spur her on and trigger her thinking process. David more than the others seeks confirmation, he wants to be able to show off his proficiency. Therefore it should be a way for each student to put their own personal touch on the exercises they are doing in order to stand out. Even if Lisa needs someone to imitate to learn better, there is also a need for her to stand out and make it her own solution to a problem. This stems from her wish to fit in the group, she wants to be on par with her friends in their progress.

5.8 Second Sessions with Digitalverkstan

We continued our cooperation with Digitalverkstan and they gave us another opportunity to host a workshop with a group of students that were new to micro:bit. We took this opportunity to test new ideas we had gotten through recent user research, mainly the personas and journey mapping. A few days after the workshop we also held a semi-structured interview with one of the other workshop facilitators to gain further knowledge that might have eluded us thus far.

5.8.1 Lindholmen Workshop

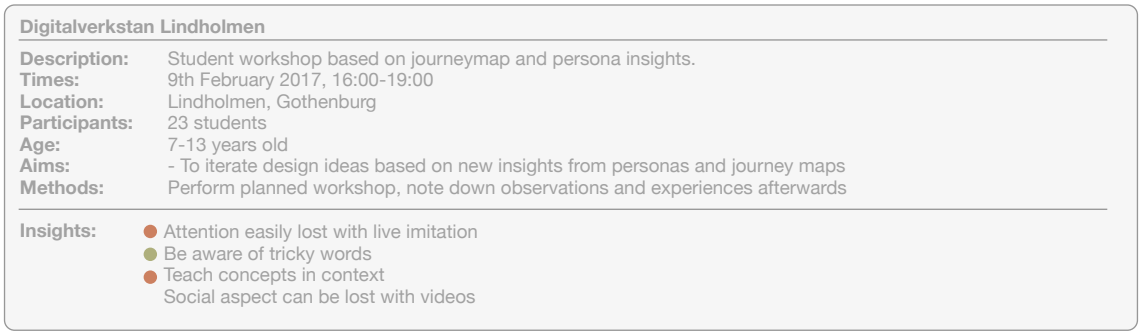


Figure 5.13: Script snippet of the workshop

After all previous user research that had been done there were some new ideas we wanted to test, with extra focus on insights from personas and journey mapping. We got in touch with Digitalverkstan, which we had done workshops for earlier, and asked for an opportunity. We were granted to host a two hour long workshop with around thirty students with no earlier micro:bit experience, although they were familiar with programming, just as we had wished our new test group to be. Ages in the group ranged between seven and thirteen, but we still prepared material for our target group age as it would not make much of a difference for beginners.

We began with a detailed presentation to introduce micro:bit and programming. The presentation included a segment with a video that was there to inspire them, and a slide regarding debugging. We tried to give them a purpose as to why programming is important as well as show the cool things you can do with a little code, and also prepare them for errors and struggling as a part of the process and something everyone faces in an attempt to give them more grit. Unfortunately none of those really got any foothold with the students. We suspect that the video was a little out of context and poorly presented by us, they might have had a hard time imagine what is possible before they even got their hands on a micro:bit in the first place. Debugging is probably better suited as a subject for later workshops, or as it surfaces during class, as it is also hard to grasp out of context.

We continued with co-coding. We chose not to hand out any micro:bits beforehand to ensure we had their full attention when showing how to upload code and program a simple name badge. We thought that the "how to" when uploading the code would stick since everyone was listening and were not distracted with their micro:bit, but we still needed to help many students with that after the co-coding. This led us to believe that a mix of showing and doing is the best fit to make such information stick. Extra attention was paid to explaining how the blocks work and how they fit together, as this was something we had learned from before that could pose a problem.

We had prepared self-instructing material, similar to what we used in London, and thus we were satisfied with just showing the students how to upload and make a simple program during the co-coding. They got hold of the exercises through a tinyurl just as we had done previously, but there were no choice this time, and only block exercises were presented. As before the slides contained videos from Makermovies.se with tutorials, instead of challenges to accompany the videos we chose to give them the necessary blocks to complete the exercise. We did this as a backup plan if there were to be any issues with the internet connection, but also since the students participating were beginners so we saw no need for challenges at this stage.

5.8.2 Interview with Facilitator

After our last workshop with Digitalverkstan we got in touch with one of the other facilitators. Anton had facilitated numerous workshops in programming for primary

school students, mainly Scratch but also micro:bit, and thus acquired experience that were of interest to the thesis as a second perspective. And as a designer he was also able to provide insights to patterns and potential solutions. We invited him to a semi-structured interview to investigate if we could extract new insights that would be of importance for our work. This session yielded some new ideas whilst confirming some we already had. According to Anton it seems to be important to create exercises that are appealing to entry level students but that can also be interesting for more advanced students. This also applies outside of the one-time introduction workshop, and is something we had pondered before. He told us that he usually starts with a "defining" exercise, were he could assess how much guidance the students would need and then adjusts the workshop according to that. We suspect that this method would also be a good way for probing the class progress during lectures as well.

If he was able to hold workshops with the same class for a couple of successive sessions he proposed beginning with a purely computer free workshop. It would consist of the students writing down instructions for some simple task, this will teach them the nature of ordering instructions and being detailed since the computer cannot think for itself. This made us investigate how one would begin teaching about micro:bit and programming for beginners without having to involve physical computers.

5.9 Workshops at Västergårdsskolan

We hereby describe the preparations and executions of four linked workshops at Västergårdsskolan on Öckerö. As previous workshops had been one time opportunities only, we were here interested in studying the progression of a class without any prior programming knowledge over a series of workshops. As the whole series was regarded as one experiment, there was an overall preparation, however there was still room for local iterations of evaluation and design between each individual workshop.

5.9.1 Preparation

Previous workshops had exclusively been focused on studying single first time workshops. Next we were interested in studying a series of multiple workshops, for being able to see the students progression across multiple sessions. The previous interview with another workshop facilitator inspired us to explore the possibilities of creating workshop exercises following a more coherent theme. Previously we had put little to no effort into creating coherence between different sections of workshops. Feedback from this second workshop facilitator indicated that incoherence might lead to confusion among the students and that one of his primary aims when designing workshops is trying to design for a good flow through different sections of the workshops. We wanted to test this idea of linking exercises and workshops together into a more coherent theme for this upcoming series of workshops.

A second ambition with designing these workshops was to not use any new tricky

words without being able to thoroughly link them to existing knowledge and explain them in easy terms already familiar to the students. The creation of personas had shown us that there was a rather large span in students initial knowledge and expectation to working with programming. It had also been seen that students who had never worked with anything similar to the programming platform Scratch, had a much harder time diving straight into working with micro:bit. We therefore had the idea to provide the whole class with an initial experience that would level the field in ensuring that everyone had access to the same baseline level of knowledge required for building further knowledge. This unifying experience was envisioned to introduce the students to basic programming concepts without mentioning new tricky words. This experience was envisioned to act as a future aid for the teachers to help reference new concepts back to this common experience. We therefore tried to design it in such a way that it would make future learning of basic programming concepts easier.

The programming concepts chosen to be introduced through this game were inspired by the CAS Barefoot website(31) and based on observations from previous workshops. Sketches of this unifying experience turned into the design of a game in the physical space. Mentioning this to our client supervisor, he recommended looking into physical programming games already created by kodboken.se(38), Botrace(39) and the Swedish tv-show Programmera Mera(40). These helped inspire the creation of further cardboard prototypes of our physical game.

5.9.2 First Workshop

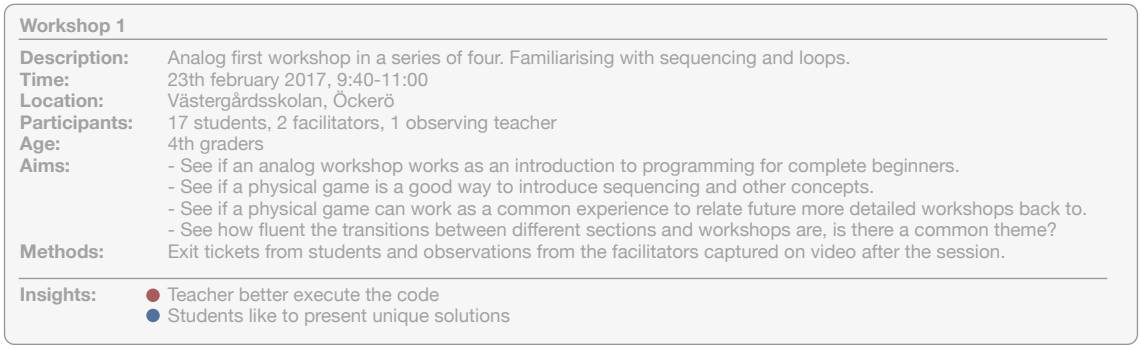


Figure 5.14: Script snippet of the workshop

Having more than a single workshop at our disposal with this class gave us the courage to try running a first fully analog introduction workshop without even introducing the micro:bit nor computers. This was thought to help students familiarise with basic programming concepts through the facilitation of a game in the physical realm. The game was inspired by kodboken.se(38), Botrace(39) and the Swedish tv-show Programmera Mera(40). We prepared it by printing and laminating number tiles, creating instruction boards out of cardboard and instruction notes out of post-its as shown in figure 5.15.

Figure 5.15: Students placing the blue instruction blocks in a sequence on the board

As students were divided into groups of four and we asked each team to present their solutions, it soon became apparent that the groups were eager to be first to present. In some cases teams changed their solution just to be different from the ones that had already presented, even though both solutions were correct. And the teams who got to present last, whose solution mostly had already been shown, showed much less interest in presenting. This made us think that the students might get a thrill out of being able to present unique solutions to exercises rather than just presenting the same ones as everybody else.

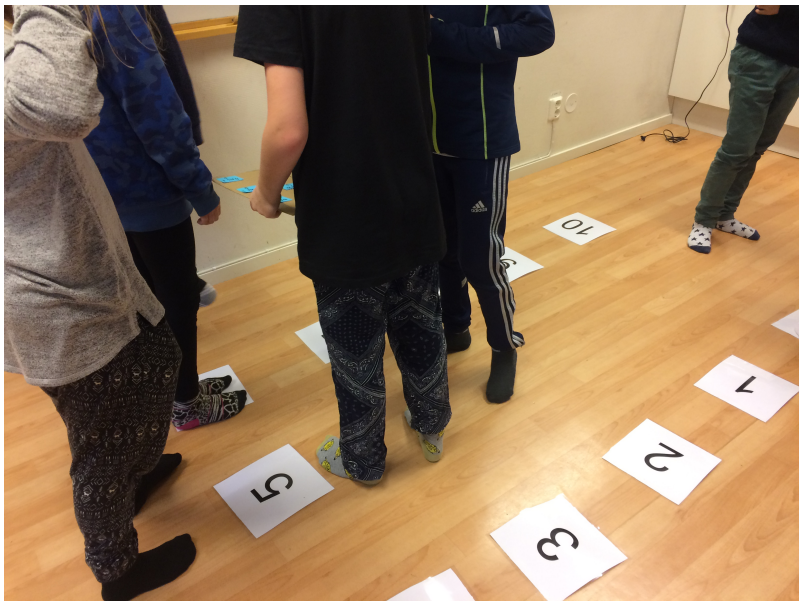


Figure 5.16: Students testing on their feet testing their programs

When students presented their solutions they initially had one student read the sequence of instructions and another student from the same team execute them on the playfield, as a remote controlled robot. This however turned out to not work as expected, as the person reading the instructions read them wrong in favor for the team. Therefore we found it better having the facilitators take over the role of being the executor of the students instructions.

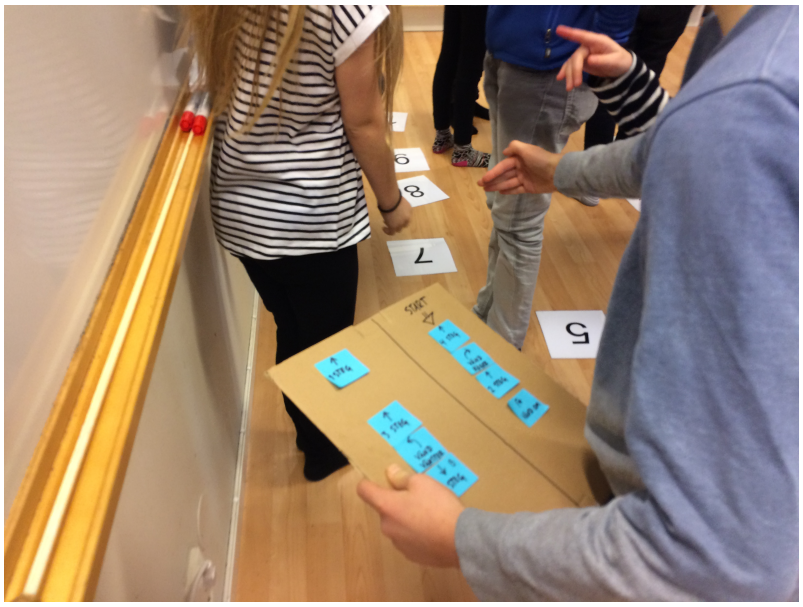


Figure 5.17: One student executing the code by giving commands

As we relied on our memory for remembering the layout of the playfields, the zig-zag playfield was remembered wrong and we accidentally gave the students an impossible problem to solve. Here we saw that they gave up after 10 minutes of struggling, but were relieved when we told them that we had messed up and that it indeed was impossible to solve. The initial idea was to only run the first workshop without any technology. After running the workshop however, we realized that the two topics we had introduced, sequencing and looping, were not enough for creating any interesting micro:bit exercises. Most interesting exercises that we wanted to introduce, like the step counter for instance, consist of a combination of basic programming concepts like sequencing, looping, variables, logic and so forth. We therefore decided to extend the analog game to cover the second workshop as well. Hence allowing the students to familiarize with all the concepts we considered to be relevant, before these concept were introduced in any micro:bit exercises.

The exit tickets asked the question “what is programming?” something that 80% of the students could answer satisfactory by the end of the workshop. For being able to say anything about how the workshop affected this knowledge however, we would have had to ask this prior to the workshop as well, something that we unfortunately did not do.

5.9.3 Second Workshop

Workshop 2

Description:	Continue familiarising with the new concepts of randomness, logic and variables through the analog game.
Time:	27th february 2017, 9:00-10:00
Location:	Västergårdsskolan, Öckerö
Participants:	17 students, 2 facilitators, 1 observing teacher
Age:	4th graders
Aims:	- See if we can find an appropriate difficulty level for students new to programming - See if we can get the programming concepts of if-statements and variables across through an analog game
Methods:	Exit tickets from students and observations from the facilitators captured on video after the session.

Insights:

Figure 5.18: Script snippet of the workshop

The second workshop, in the series of four, was decided to be a fully analog workshop as well, to continue with familiarizing the students to the basic programming concepts. Previously they had been introduced to sequencing, loops as well as some level of debugging through the trial and error problem solving approach. Next we wanted to introduce game elements that could help them understand the programming concepts of randomness, logic and variables. This was prepared through having playfields that split into two different directions with different colored goals. Which goal they were supposed to go to was dependent on the outcome from drawing a colored ball from a bag at random. Each group was also provided with a selection card to create different instruction lists depending on what color the ball was. These two elements were expected to make it easier to later understand the programming concepts of randomness and logic selection with if-statements. For variables we introduced a high score counter card, as well as two instruction notes that allowed each team to add the number of the tile, their player was currently standing on, to their high score. The game was to complete the game as previously, but gain as many points as possible at the same time.

Figure 5.19: Split playfield for the logic exercises

This time we made sure to prepare and test all playfields in advance. The last

playfield was simply a straight line to make it easier for them to focus on the new concepts of variables, or high score counters as we called them at the time. Not mentioning the word variables was a conscious attempt to not scare them with new words, but rather build on to something that they might already be familiar with. This went well, and in the end we even allowed them to make use of the loop blocks introduced in the previous workshop. We were aware that they with these blocks would be able to "hack" the game and eventually get ridiculously large high scores, but only if they had understood the concepts of them. We were therefore very happy to see that they indeed did do this and managed to get infinitely large high scores by looping something over infinity. The students were happy as they thought they had cheated the game, we were happy that they had understood the concepts, and since they had now used a loop repeated infinitely, we had a convenient leverage to teach the fundamental programming concept of a forever loop.

Figure 5.20: Straight playfield for high-score exercises, blue selection card and red counter card seen on the floor

5.9.4 Third Workshop

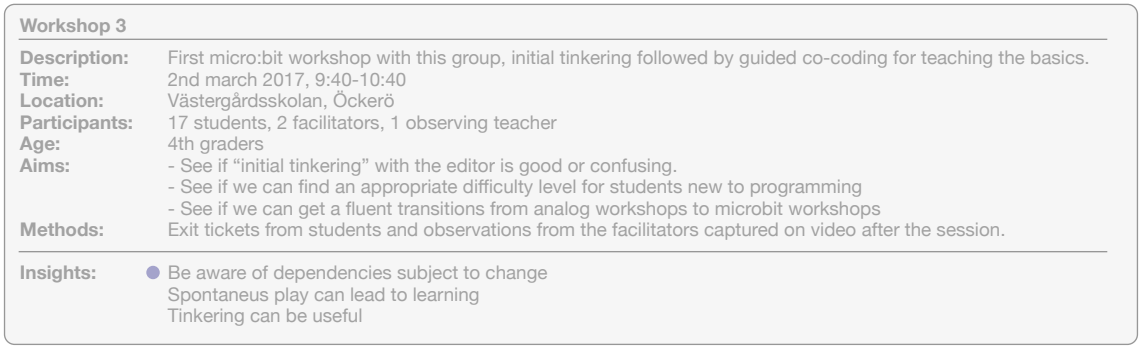


Figure 5.21: Script snippet of the workshop

This was the first workshop session where we introduced this particular group of students to programming with the micro:bit. As we previously had seen that classes who had no prior programming experience had a much harder time getting into working with the micro:bit than those who had worked with Scratch for instance, we wanted to see if our analog game had prepared this group for a smoother transition to working with the micro:bit, than those who had no prior knowledge. Our subjective observations suggest that the class who had gone through the analog workshops had less trouble working with the micro:bit than those who did not have any prior programming encounters.

This initial workshop was intended to be focused on imitation and familiarisation rather than problem solving. After the students had a short introduction to the micro:bit editor we allowed them to have 10 minutes where they were allowed to freely tinker around with the editor. After 5 minutes we asked each group of two to switch who sat at the computer, to ensure that both students got to spend time tinkering. Our hopes were that they here would have time to get outlet for any curiosity and familiarise with this new element. The time was limited in an attempt to prevent frustration to arise, as there were no instructions. The students showed interest in clicking around in the new interface and tried things out without any apparent hesitation. We did this before handing out the actual micro:bit hardware in an attempt to allow the students to focus on the editor without the distraction that often arises when we start to hand out the micro:bits and cables.

Next we handed out the micro:bits and cables. We showed on the projector how to create a simple name badge program and how to transfer it to the micro:bit.

Next we showed them how to make a step counter and related the notion of variables back to the analog workshop game where they used high score counters. It felt useful to have this reference when talking about variables and they seemed to understand. They were however merely imitating, so it was not really possible to test their understanding at this point. An interesting thing that happened was that after the students had created their step counters, they started running around playing a self

invented game of who could shake their micro:bit the most to get the highest number of shakes shown on their display. At first we thought that we might had to interrupt this game as it did not seem to be very productive. However we allowed them to have some fun, and to our surprise some of the students went back to their computers modifying their micro:bit programs to add not one but thousands of numbers with each shake. This "hack" made them superior in the game the students had invented, and some called it cheating.

The analog games made for the previous workshops had been designed with an intentional color coding in mind, to match the colors of corresponding parts of the micro:bit editor. Variables were bordeaux, the logic cards were blue and so on. The same day as this third workshop however, the colors of the editor had been changed. This made us think about the dependencies that created teaching material can have to software that unexpectedly can change due to updates.

5.9.5 Fourth Workshop

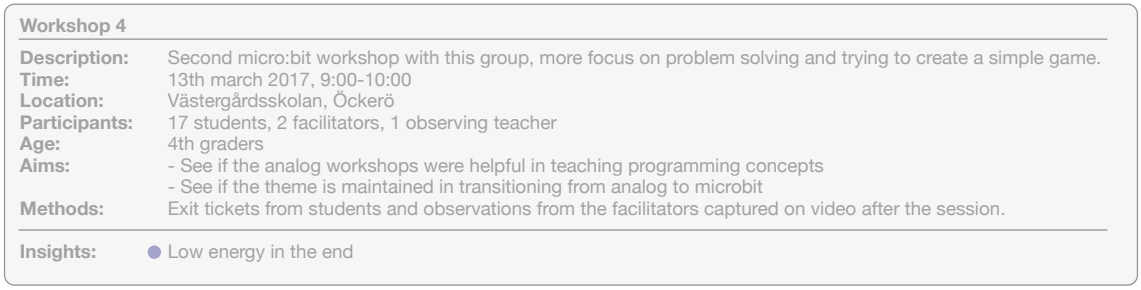


Figure 5.22: Script snippet of the workshop

The fourth and concluding workshop with the students at Västergårdsskolan was a continuation of the previous micro:bit workshop where the students had been introduced to the block editor and where the basic programming concepts had been linked to the prior experiences of the analog game workshops. Now as the students had been introduced to the basic programming concepts within the framework of the block editor, we wanted to see if they were able to solve problems on their own with less guidance. We tried teaching them the problem solving approach of first defining the desired program behaviour, before starting to think about the actual implementation.

To illustrate this way of working we co-coded the dice program together with the students. The co-coding was here also used as a probing tool for us to see how much the students remembered from the programming concepts that had been taught in the previous workshop. Our subjective experience was that the students showed a lower level of knowledge than previous workshop had made us believe that they had. Next the students were asked to describe the behaviour of a rock paper scissors game. From agreeing on a behaviour the students were asked to create such a game. It soon became apparent however, that the free level of problem solving involved

in this exercise was rather overwhelming compared to the previous workshop where they simply were asked to imitate what was done on the big screen by the facilitators. The ambition to teach problem solving through separating desired behaviour from implementation, was also not considered to be very successful, but rather confusing for the students.

From the previous workshop we had seen that the students enjoyed playing the self invented game of shaking the micro:bit to gain a high score. That inspired this workshop to be focused on creating the rock paper scissor game. It turned out however that the students still preferred to go back to their own game from last workshop, and that they did not show as much enthusiasm about this game that we had imposed onto them. It should also be mentioned that there was a gap in time of approximately one week between the last two workshops.

5.10 Insight analysis

The last step in our process was to summarize and analyze insights from all the previous activities in an attempt to look for patterns and more abstracted insight. This was done through the affinity clustering method and resulted in one main insight that we came to call the scope of autonomy model. This was accompanied by a set of other considerations that we found useful for anyone to ponder who would be interested in conducting micro:bit workshops of their own.

We were initially appealed by the idea of trying to deliver one big unified result encompassing all of the insights that we had obtained throughout the project. In some sense that was what we ended up doing with the scope of autonomy model, but there were still many minor insights encountered along the way that did not directly make it into this model. Still we considered many of these insights to be valuable to teachers who would want to conduct their own micro:bit workshops at some point. Hence we discussed different formats for including these in the result. We thought of delivering the more practical tips, as a troubleshooting appendix, but eventually settled for simply bundling all insights into a collection of considerations for anyone who would want to work with the micro:bit. This bundle therefore consist of some considerations that were aggregated from multiple insights from along the project timeline, whereas others were single insights that we simply found useful to share. We chose to communicate these topics as considerations rather than guidelines or suggestions, as they had been identified through subjective qualitative means, and lack any formal proof of being generally applicable.

Besides the scope of autonomy model there were five further topics for consideration that were aggregations of multiple insights from throughout the project's timeline. These topics were: analog workshops, basic toolbox, terminology, technical pitfalls and co-coding.

5.10.1 Affinity Clustering

As our process was iterative, each iteration consisted of the scripting and executing of a workshop intervention. These interventions were also followed up by smaller local evaluations, to catch any observations or new insights that could feed into the following iteration cycles. By the end of the project these local evaluation insights and observations were finally summarized and aggregated using the affinity clustering method. This was a long and tedious process where we had to rearrange and juggle the post-it notes many times until we felt that redundant duplicates had been merged and a satisfying structure had appeared. The process of rearranging the post-it notes could at times make us feel as if we had hit dead ends, and the feeling of being stuck could get rather frustrating. At these times we tried to mix things up, by scrambling the post-it notes and trying to map our data to theoretical models that we previously had found in our literature study. For instance we tried to map our data to the skill, will and thrill mentioned by Hattie and Donoghue(22). Their model also mentions surface and deep learning as two different phases of learning, something that we also could discern in our data.

Figure 5.23: At the beginning of the clustering

We had the impression that some of the insights were more substantial than others. Some were more on the practical side about simple troubleshooting. We therefore excluded these rather trivial insights from our affinity clustering initially, with the intention to add them as a troubleshooting appendix later. After feedback with our client supervisor however, he suggested that we still should include these trivial insights but rather look for abstract patterns within them.

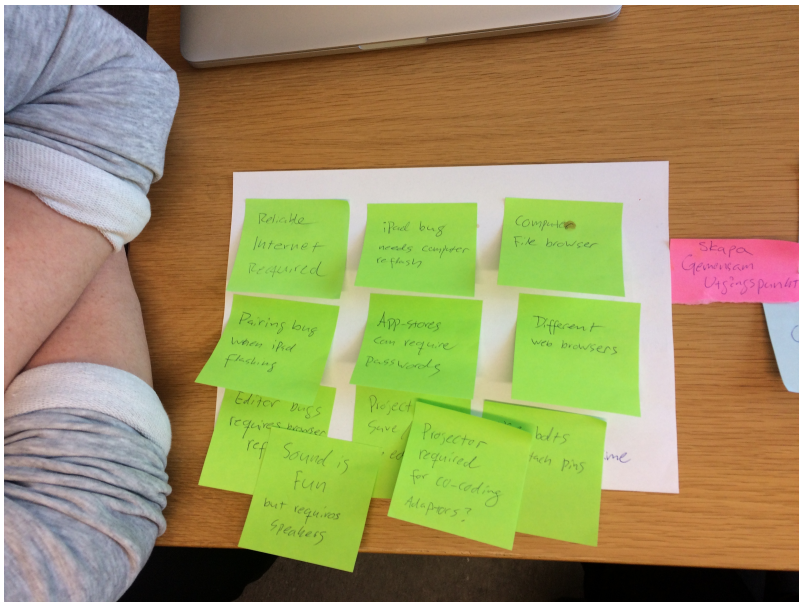


Figure 5.24: Set of more practical insights

An initial ambition was to find parameters that could affect workshops, and depending on these input parameters give teachers output suggestions on how to plan and run their workshops. This was dropped however as it did not map very well to the outcome of our affinity clustering. What did attract our attention through this method however was the fact that many various observations seemed to relate to the amount of freedom the student was given in their exercises. This later evolved into a model about the students scope of autonomy and regarded as this project's main contribution. Still there were other groups of aggregated and single insights that will be presented as considerations for working with micro:bit.

5.10.2 Scope of Autonomy Model

From the affinity clustering of all the insights and observations from the individual workshop interventions, we started to see an emerging pattern relating to issues with balancing the level of freedom given to students. As some observations were that students did enjoy being able to explore and do things in their own way but at the same time we saw some issues when students were given too much freedom as this could lead to confusion and a feeling of being overwhelmed.

We were inspired by the idea of design spaces, and imagined a space where moving to a new point was equivalent to making a design decision. And we saw ourselves as facilitators as holding the students hand, guiding them through some of these decisions towards the goal. But at other times allowing them to be free enough to make decisions for themselves, as we saw making active decisions as an important part of the learning process. Viewing learning in this way also made it interesting to think about the many situations where there are more than one correct way to complete an exercise. In these cases we as facilitators have given the students

enough freedom to choose any out of the sometimes infinite possible designs that could satisfy the exercise.

We realised that we as facilitators never really had reflected on the amount of freedom we had given students through various exercises. Sometimes we gave them no freedom, like when telling the students exactly what to do and to simply ask them to imitate. Other times we had given them full freedom, by asking them to come up with their own ideas for a program they would like to build. We realised that we wanted to make an active decision in the creation of an exercise, about how much freedom was given to the student within that exercise. Consequently the design decisions that would not be available to the student would have to be made by the teacher.

The freedom that is given to a student in their design decisions, we came to call the amount of autonomy that they were given. As we saw these levels of autonomy as increasingly larger areas to explore within the design space, we chose to call them scopes of autonomy. We chose to visualise these scopes as circular zones that would get increasingly larger with more autonomy, representing that the student's autonomy then covers more decisions. This representation was also chosen to indicate that each new level of autonomy also includes the previous levels. The area outside of the students scope of autonomy was chosen to represent the choices to be made by the teacher.

In parallel with this we tried to brainstorm and identify the different levels of autonomy present in working with the micro:bit, starting with the least amount, which is no autonomy at all. This means that the students were not given any choices. This was discussed to be true for traditional lectures as well as pure imitation exercises. From our experience this was not desirable, and we found it more effective to teach programming through exercises that at least allowed for some level of autonomy, even for the very beginners.

The first real level of autonomy that we identified in relation to micro:bit exercises, was in allowing students to make smaller customisations to a predefined design. In the case of a simple "hello world" program, this could mean allowing the student to customise the text string to something else than "hello world". Hence a customisation is not something that alters the behaviour of a design, but rather allows the student to locally modify specific point of interest that has been preselected by the person designing the exercise.

The next level of autonomy that we identified was the one relating to the solution procedure of an exercise. This level of autonomy relates to what subparts of a solution to tackle in what order. When this level of autonomy lies within the student's scope, the student is free to chose the order in which to create the solution. When the solution procedure does not lie within the scope of the students autonomy, they are asked to follow a stepwise procedure instructed by the teacher. In the case of creating an animation, this could relate to the difference in starting with drawing the

desired animation and then figuring out the best timing between frames, or doing it the other way around.

The third level that we identified, was concerned with the design that a student makes to complete an exercise. This is a rather interesting level of autonomy, as setting the students scope of autonomy to encompass this level means that the teacher no longer knows what the final design will look like, as it is up to the student. From the analogy of design spaces, this means the teacher no longer knows what destination to direct the student towards, but instead the teacher has to guide the student to find his or her own destination that satisfies the assignment. In contrast to a scope of autonomy that only encompasses the level of customisation, a scope that encompasses the level of design allows for completely new design solutions to an exercise, and not only the modification of predetermined placeholders.

The fourth level of autonomy that we identified was related to block selection. Blocks are the building pieces that are used to create a design. When this level is not encompassed by the students scope of autonomy in an exercise, it means that the teacher has predetermined what blocks the student should use to create their design. This level of autonomy has two parts. The first of which relates to the type of blocks and the second one relates to the quantity of blocks. For instance the teacher can give an exercise where the students are asked to create an animation on the micro:bit using any number of blocks of the "loops" and "show LED" variety. Another exercise could be to make a step counter using only four blocks in total. These two examples illustrate that an exercise can be created in ways where the students scope of autonomy only encompasses one of these two block selection levels. Likewise none of them can be encompassed, which means that the teacher decides exactly what blocks ought be used. And lastly when both of them are encompassed by the student's scope of autonomy, it means that the student is free to create a design out of any block type or quantity, as long as it satisfies the assignment.

Lastly we identified an autonomy level relating to the very assignment itself. This relates to decisions about the topic and aims of an exercise. In the case where this level is not encompassed by the students scope of autonomy, the teacher defines what the student ought to do in order to complete the exercise. When this level is encompassed by the students scope of autonomy however, it is up to the student to make the decisions about what the exercise is going to be about. Initially we only associated examples of these kind of exercises with higher educational projects, similar to the one you are reading right now. We realised however that some of our earliest workshop exercises, where we tried to encourage students creativity by telling them to "create whatever they wanted", also could fit into this level of autonomy. As having to create your own assignment basically is the same as having to come up with an original project idea. For being able to handle this level of autonomy however, our experience is that the students needs to have reached a rather high level of experience and be comfortable with making all these kinds of decisions, or they might run the risk of feeling overwhelmed.

5.10.3 Co-Coding

Co-coding relates to the activity of having the teacher stand in front of the class, screen sharing his or her computer screen on a projector and solve programming exercises in a dialogue with the class. This was found to be a useful hybrid between pure presentations and having students solve exercises on their own. As presentations were considered to be good for introducing new knowledge, but that it was undesirable to put the students in a rather passive seat. On the contrary, individually working with exercises, was considered to be more active but not ideal for introducing novel information. Co-coding hence evolved as a middle path between these two approaches. Our experience was that it was a useful method as it allows the teacher to probe students current skill level with questions and directly adapt the level of guidance to that level. This method was partly inspired by Hattie and Donoghue(22) mentioning of teaching learning strategies in context rather than separately. Similarly we saw co-coding as a way to teach new concepts, like the need for variables for instance, within the context of an exercise, rather than as a separate presentation.

5.10.4 Technical Pitfalls

Technical pitfalls refer to practical issues that have been identified to risk obstruct or fail the execution of a workshop. As it has been observed that students can enter states of indifference or dejection if these issues take up too much time, it is suggested to take measures and attempt to prevent them from arising in the first place. Three specific issues that have been observed are relating to: app-store passwords, internet connection and pairing mode bugs.

5.10.4.1 App-Store Passwords

In the context of running a micro:bit workshop with iPads, the students are required to download the micro:bit app from the app-store. This requires an app-store account and as some schools prefer to control what apps are being installed on their iPads, some schools protect their accounts with passwords. It is therefore recommended to obtain these passwords well in advance and work out a good way for these apps to be downloaded in class, or possibly even downloading the micro:bit app to each individual device in advance.

5.10.4.2 Internet Connection

Unreliable internet connection was seen as another frequent source of frustration. As the micro:bit editor is run through the web browser reliable internet connection is required throughout the workshops. In the cases where internet was slow or the connection was dropped occasionally, our experience was that a lot of the time went by helping frustrated students troubleshoot rather than teach them about programming. In the cases where workshops rely on online material, such as instruction videos, the internet bandwidth plays an even more noticeable role, as video can be rather bandwidth heavy.

5.10.4.3 Pairing Mode Bugs

Lastly there were some issues identified regarding the pairing of micro:bits to iPads. Firstly there is a certain procedure that is required to initially pair a micro:bit to an iPad. This procedure can be rather tricky at first, as it requires the student to press three buttons in a certain order and remember a series of six numbers shown in a rapid sequence. Our experience is that students best understand this by first being shown a whole pairing procedure and then try it themselves. Having students imitate this procedure in real time is not recommended, as it can lead to disorder in the class. Next issue we encountered was presumably a bug relating to sending a program (flashing) from a mobile device to a paired micro:bit. We found that this process required the micro:bit to be put into pairing mode again, despite that this is not mentioned in the documentation. Our experience from talking with teachers also shows that this is an issue that can hinder entire workshops. Lastly there have been a few rare occasions where micro:bits were impossible to put into pairing mode. This bug was resolved by simply having a computer nearby and flash any type of program from the computer to that micro:bit via usb-cable. This way the micro:bit gets reset and can be paired with an iPad again.

5.10.5 Basic Toolbox

Basic toolbox relates to a set of fundamental programming concepts that were found useful for students to have been introduced to, prior to working with programming exercises. The programming concepts that we found useful for this were: algorithms, loops, randomness, logic, variables and debugging. These concepts were the ones we had focused on conveying trough the analog workshops at Västergårdsskolan, which in turn had been inspired by the CAS Barefoot website(31). Algorithms refer to the sequencing of instructions to reach a certain desired behavioural outcome. Loops refer to the fundamental programming concept that allows certain instructions to be repeated multiple times. Randomness refers to the basic programming function of a random generator, which is frequently used in many types of programs. Logic refers to another fundamental programming concept of statements that are verified to be either true or false, like in if-statements for instance. Variables are a vital part of programming and the concept of an addressable data entry that can be recalled and changed at a later time. Debugging refers to the mindset and activity of expecting errors in your code and be willing to pursue and fix them.

5.10.6 Terminology

Terminology is referring to the words used when teaching programming. As there are many new words and concepts that might be intimidating for students at first, we found it useful to initially avoid using words as variables, but rather attempt to convey these through words and concepts that already are familiar to the student.

5.10.7 Analog Workshops

From the insights about a common point of departure and the needs for a basic programming concept toolbox, evolved the idea of a common shared experience that the whole class could have as a reference for future learning. These so called analog workshops were seen to provide students, that were lacking prior experience with Scratch, enough programming knowledge to reduce the hurdle in starting to work with micro:bit. Analog workshops relate to the idea of not starting out with introducing students to the technical programming platforms straight away, but rather take one or a few sessions where the programming concepts are practiced through other means.

5.10.8 Other Considerations

Furthermore there were also eight single insights from workshop interventions that were added to the list of considerations for working with the micro:bit. These topics were related to: tinkering, self instructing materials, stupid computers, end on a positive note, text based instructions, video bubble, editor navigation and awareness of dependencies.

5.10.8.1 Tinkering

Tinkering is allowing students to freely familiarise with new content before starting to work with exercises. This explorative approach without any goals or objectives was seen as a way for students to get outlet for any curiosity that might arise as they are introduced to new technology, platforms or concepts. To prevent frustration or confusion to arise, it is recommended to keep this initial tinkering short in time.

5.10.8.2 Stupid Computers

Stupid computers refers to making it clear to students that computers are simply following instructions and should not be considered as intelligent per se. In some cases we have seen students who expect that computers are smart just because they are computers. Clarifying this initially can potentially prevent some of these misconceptions.

5.10.8.3 Text Based Instructions

Purely text based instructions might not be the optimal way for conveying exercises. Giving instructions through other means can potentially be more successful. As we have seen students ignore printed papers with instructions that have been handed out. Conversations with students also showed that they found written instructions incomprehensible, and said that they were glad we as facilitators were there to answer their questions. Other means of giving instructions could also be through video.

5.10.8.4 Editor Navigation

Editor navigation refers to the need to understand how a certain programming editor works and how to navigate it in order to use it. It is easily overseen and can be considered trivial by someone familiar with it, nevertheless is it important for a first time user.

5.10.8.5 Self Instructing Materials

Self instructing materials such as instruction videos, have been seen as potentially useful in cases where student group sizes exceed the number of students that the facilitators are able to provide help to.

5.10.8.6 End on a Positive Note

End on a positive note refers to concluding workshop sessions with an exercise that leaves the students in a positive state of mind, rather than leaving them confused or frustrated. Struggling with exercises might be important for the development of grit, but in our experience it is better to place these harder exercises in the middle of a session and allocate the end for easier ones that allow students to feel successful. From a facilitator point of view this means planning the timing of the session well, and never try to cram any exercise in the very last minute, just because you want to convey something that might not really have gotten across. In our experience the energy level of the class is usually rather low in the end, and trying to force last minute teachings in here, seems to possibly make more harm than good.

5.10.8.7 Video Bubble

When using instruction videos as teaching material for an entire class, one ought to be aware of the potentially negative effects this might have on social aspects of the group. As a whole classroom full of students watching different videos on their computers will get rather noisy, headphones are recommended. This however also has the effect of isolating students into their own little video bubble. This might be positive for some, in terms of concentration, but it also removes many of the interpersonal social interaction that can be positive in a group.

5.10.8.8 Awareness of Dependencies

When creating teaching materials dependent on software, one needs to be aware that it can be changed in future software updates. For instance if one creates a set of instructions on how to navigate an editor that in detail refers to certain buttons, the names of these buttons may well be changed in future software updates. To avoid the risk of confusing students, it is therefore suggested to continuously verify that the teaching material with dependencies is up to date with the current state of the software.

5.11 Reiteration of Result

At the time of our oral presentation our result was heavily focused on emphasizing the model, with the rest of our material, insights and considerations receiving less attention. The presentation was an opportunity for opponents, audience and examiners to provide feedback and for us to evaluate if there needs to be any changes done. Questions were raised why we had chose to withheld the other results from receiving more light, as we could claim more about the result than we currently did. Another issue was that the result was not properly divided into the right categories, basically everything except the model were put under an umbrella term we called "Considerations". This made us restructure our result to provide a more correct way of presenting our deliverable.

One of our goals was to provide teachers with materials that they could either use directly for their educational purposes, or tools recommended to use when designing materials on their own. Then we drew the conclusion that time available for teachers at different times is a factor when it comes to their ability and will to use the material. Therefore we split our result along an abstraction axis with three different levels.

On the topmost level we placed our Scope of Autonomy Model, which is the most abstract form of our result. We find that the model is aimed at those that can invest time in understanding the underlying theories behind our model in order to implement it correctly in their education, as well as when creating own teaching materials. Next we chose to separate a teaching approach that we practiced, co-coding, and named guidelines. The guidelines consisted of the basic toolbox, the use of terminology, technical pitfalls, and a set of minor considerations. This middle level of abstraction caters teachers with a little less time available but still with the ambition to create their own set of exercises or complete workshops. All before mentioned approaches and guidelines stems from our model in both form and use. Lastly we chose to create a collection of ready-to-go exercises and sets of complete workshops. This kind of material allows for quick and easy use without much time investment. Teachers could pick and chose from the exercises and workshops and then apply it directly in their classroom. All exercises and workshops were created with the guidelines, approach and model in mind.

6

Result

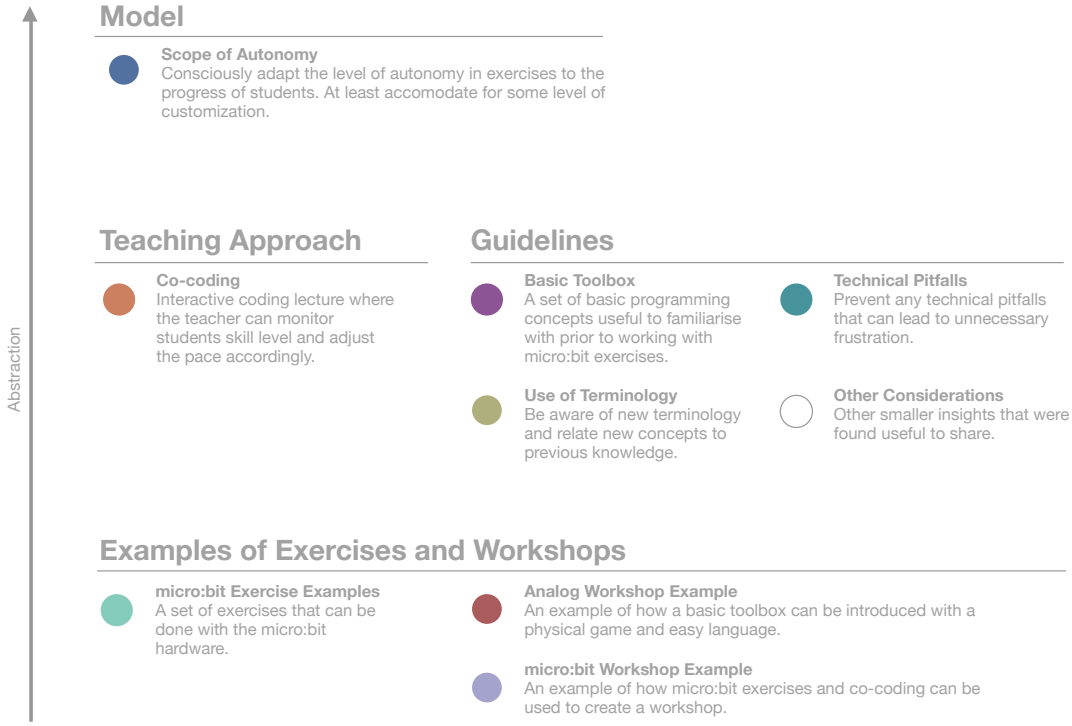


Figure 6.1: An overview of the different result parts of this thesis, mapped along an axis of abstraction

This thesis aims to answer the research question of what is important to consider when designing teaching materials with the BBC micro:bit for training Swedish primary school students computational thinking skills. To answer this question, the researchers have pursued the design goal of supporting Swedish primary school teachers with teaching materials, based on the BBC micro:bit, that help them meet the programming requirements of the new curriculum changes. Hence some outcomes of this thesis will fall into the theoretical domain, attempting to give a non exhaustive answer to the research question. Other outcomes however, rather fall into the practical domain, by providing some examples to how the design goal could be met.

The result of this thesis consists of 9 individual parts. These parts are presented in 4 result groups: Model, Teaching Approach, Guidelines and Examples of Exercises And Workshops. These groups are presented in 3 different levels along an axis of abstraction. This is to help make the result more accessible to teachers and enable different readers to find the information that is most relevant to them. If a teacher has a lot of time to develop their own material for instance, it might be more interesting for them to look at the more abstract model concerning what is important to consider when designing teaching materials for the micro:bit. If the teacher has less time on their hands however, it might be more convenient to look at the more concrete examples provided further down along the abstraction axis.

In this chapter these parts will be presented starting with the least abstract parts first, gradually moving up to the more abstract results later. Result groups that contain more than one part are given their own top level chapters, result groups that only contain one part however, have been simplified to a single chapter level for structure simplification. These results are derived from empirical research done in a Swedish primary school context with the aim to train students computational thinking skills using the micro:bit.

6.1 Examples of Exercises and Workshops

This chapter contains the least abstract result group, and is intended for teachers who might not have enough time to create their own material from the more abstract result parts, but rather want something they can put to use straight away. This result group contains 3 parts: micro:bit Exercise Examples, Analog Workshop Example and micro:bit Workshop Example.

6.1.1 micro:bit Exercise Examples

This chapter provides a few micro:bit exercise examples that have emerged throughout the project and been considered useful for the creation of teaching materials within the scope and context of this thesis. Most of the exercises emerged rather early and some of them have been refined throughout the project.

6.1.1.1 Animation

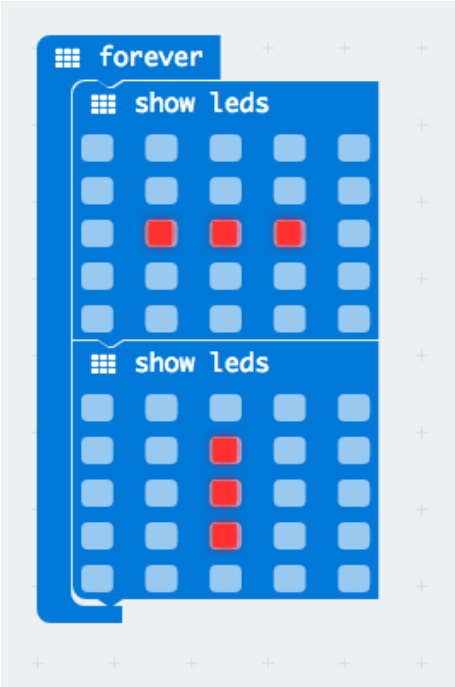


Figure 6.2: The block configuration for one type of animation

This beginner exercise revolves around allowing the students get familiar with the micro:bit in the easiest way possible. Have the students combine several *show leds* blocks and create an animation on their micro:bit.

Possible ways of building further is to be able to show different animations depending on an input, i.e a button press.

6.1.1.2 Name Badge

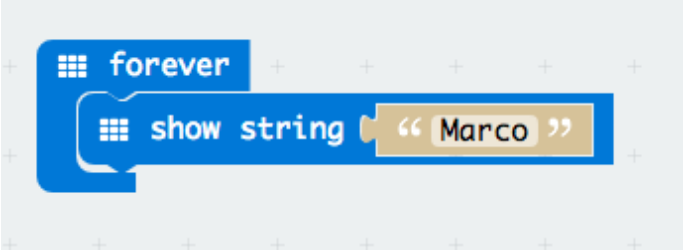


Figure 6.3: The block configuration for Name Badge example

The Name Badge exercise is a program that scrolls a text string on the micro:bit's led display.

By adding inputs to the mix we can control what is shown on the micro:bit as seen

in 6.4. There is also the possibility to send the text strings via blue tooth by using the radio blocks.

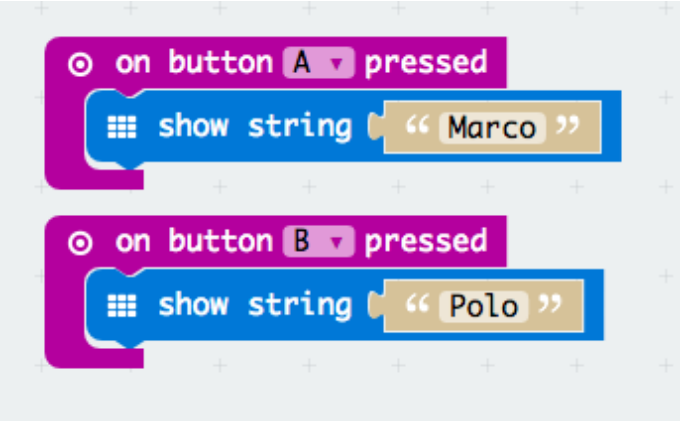


Figure 6.4: Name Badge controlled by inputs

6.1.1.3 Coin Toss

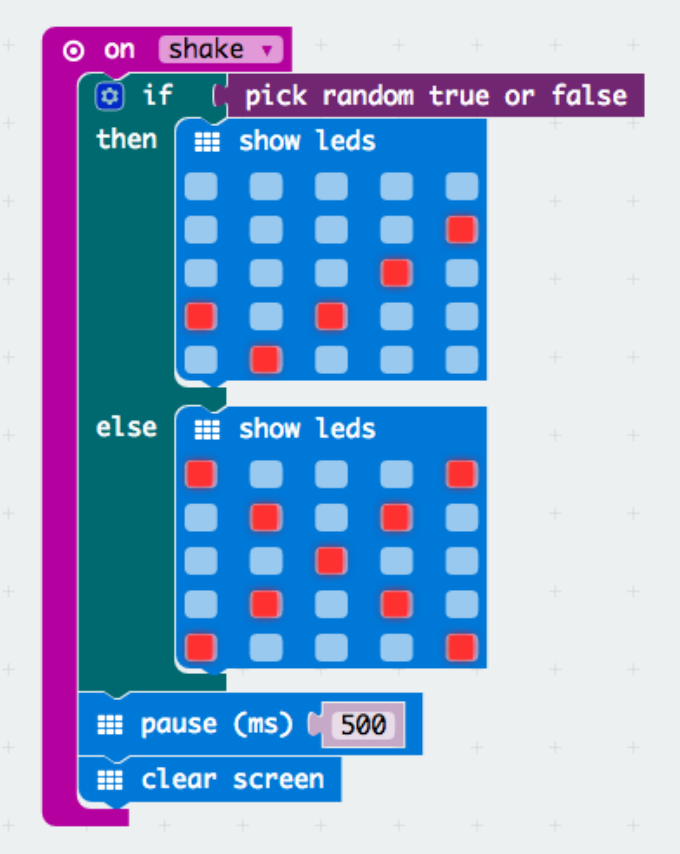


Figure 6.5: The block configuration for Coin toss

Coin Toss is a simple true or false program. It is a first introduction to booleans, which are frequently used in all programming languages and an important ingredient

in algorithms. When the micro:bit is shook, it will display one of the two led images as seen in 6.5 for half a second before showing a blank display again.

We noticed that some students fail to understand that the program is completely random each shake, to help visualize that a *pause* and *clear screen* were added. This make it easier to distinguish if there is two (or more) of the same image in a row.

6.1.1.4 Dice

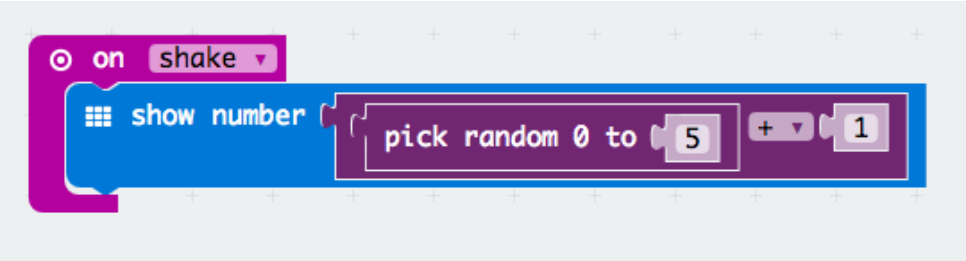


Figure 6.6: The block configuration for this particular exercise example

A variant of a Dice program. When the micro:bit is shook it will pick a random number (0-5) and add 1 before displaying it, resembling a six-sided dice.

As with many programs there is more than one solution, but we found this to be in its most simplistic form. An extra assignment could be to try and create a similar dice with a different solution.

6.1.1.5 Rock Paper Scissor

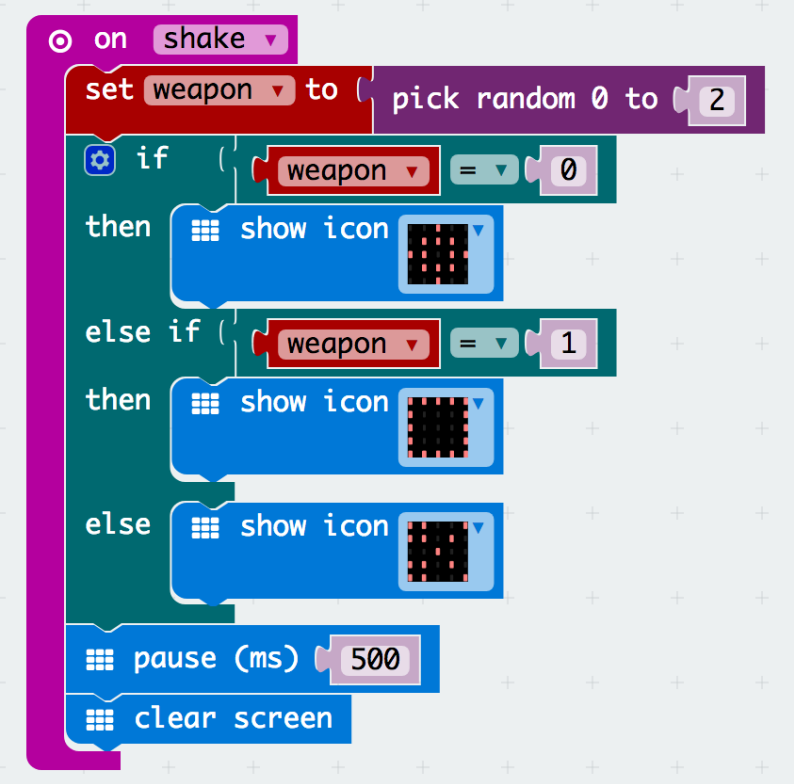


Figure 6.7: Rock Papers Scissors exercise

Rock Paper Scissors is a game that all are familiar with, and this exercise combines several key programming elements. The program starts when the micro:bit is shook, a number (0-2) is saved on variable "weapon". The code then proceeds by checking the number stored on the variable to match it with an image to be shown.

This is a great exercise as the students can test it out with their friends and compete when they are done. There are several ways to continue building upon this program, you could for example implement the game to work via radio or add code to keep track of the score for you.

Just as with the Coin Toss we added *pause* and *clear screen* to provide extra feedback from the program.

6.1.1.6 Step Counter

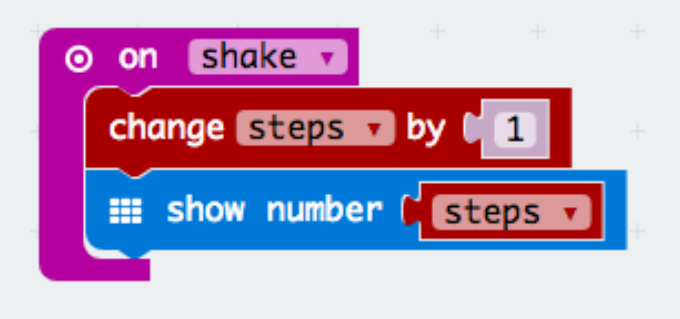


Figure 6.8: A simple step counter

Turn the micro:bit into a step counter by increasing the value of a variable "step" and displaying it every time the micro:bit is shook.

A neat extra exercise is to add a way to store the steps, and being able to display on demand.

6.1.1.7 Music Player

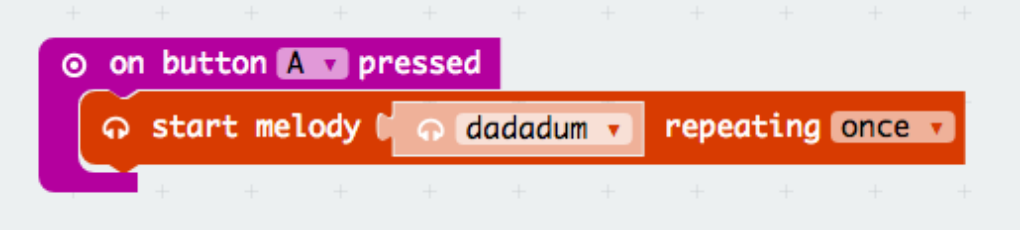


Figure 6.9: The block configuration using a preset music sample

This exercise lets your micro:bit play a melody once button A is pressed. You could either use a preset melody as in 6.9 or compose your own as shown in figure 6.10.

The micro:bit can only play a sounds if it has a buzzer connected, or if a pair of headphones are connected as in figure 6.11 using crocodile clips. If the micro:bit is still connected to a computer via USB-cable the melody will play through the computers speakers instead.

Figure 6.10: The block configuration of a self composed melody

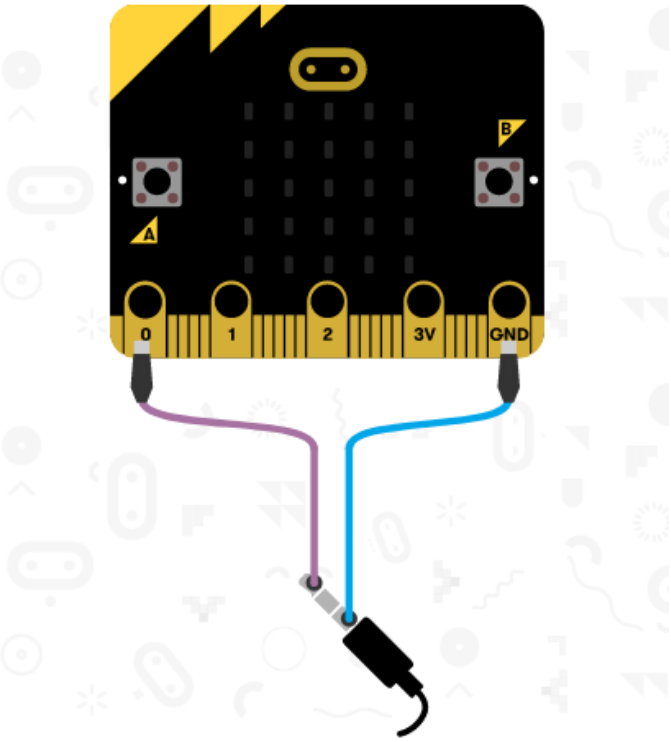


Figure 6.11: Headphones can be connected in this way for being able to hear the sounds created in this exercise

6.1.1.8 Radio Messages

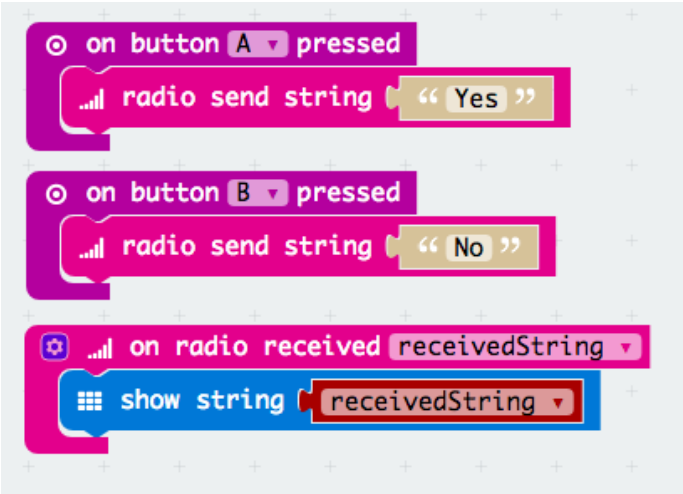


Figure 6.12: The block configuration for this particular exercise example

This exercise uses the radio blocks to utilize the built-in bluetooth features of the micro:bit in order to communicate wireless. This program sends a string "A" or "B",

depending on which button that was pressed, to another micro:bit that picks it up and shows that particular string message on its display.

It is important that micro:bits that are to communicate with each other are set to the same *radiogroup*, to be sure either add a specific block to set the radio group manually or flash the same code to all micro:bits that you want to be able to communicate with each other and it will solve automatically.

6.1.1.9 Neopixel Animation

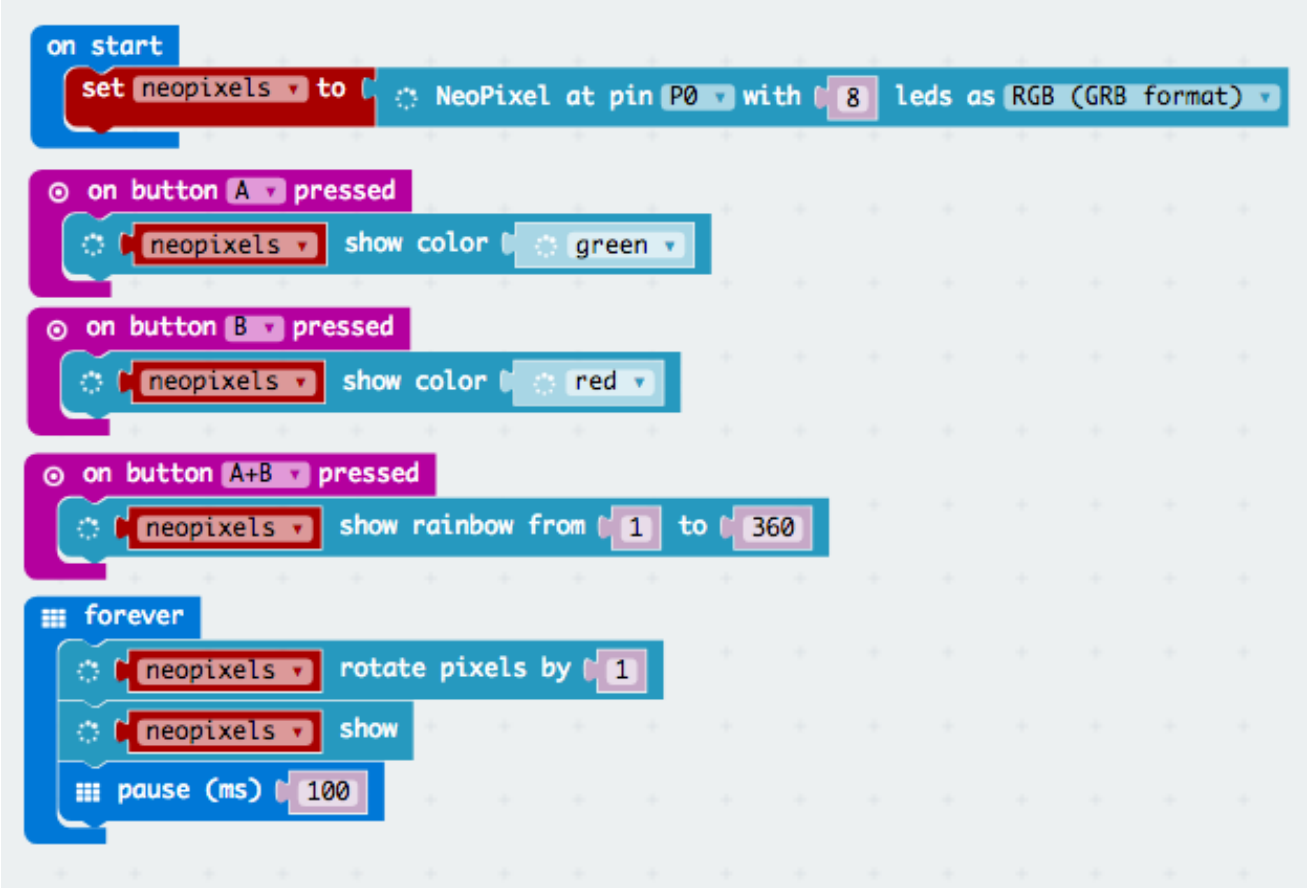


Figure 6.13: The block configuration for a typical program using Neopixels

This exercise utilize an extra package of blocks named Neopixel and hence requires neopixel strips to function. The editor will show how to connect it properly and it may look something like figure 6.14. The program sets a variable called "neopixels" to connect the output to pin 0 with 8 leds active. Depending on what button, or buttons, being pressed it will either show all green leds, all red or a rainbow.

As seen in the code we also added a block which rotates pixels, however it is only visible when the rainbow is active. *Pause* is only there to slow down the rotation of pixels and not of great importance. Also worth mentioning is that the micro:bit can

only sustain about 10 pixels before the colors start to fade due to lack of power. A way to go around this is to connect an external power source to the neopixel instead.

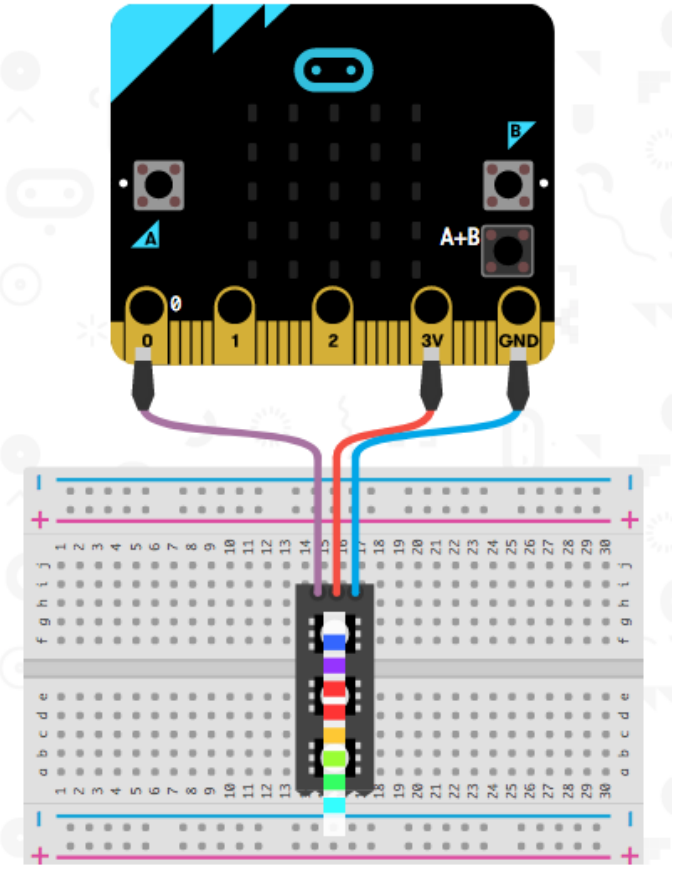


Figure 6.14: The Neopixel strip will need to be connected as shown for the code to work

6.1.1.10 Level

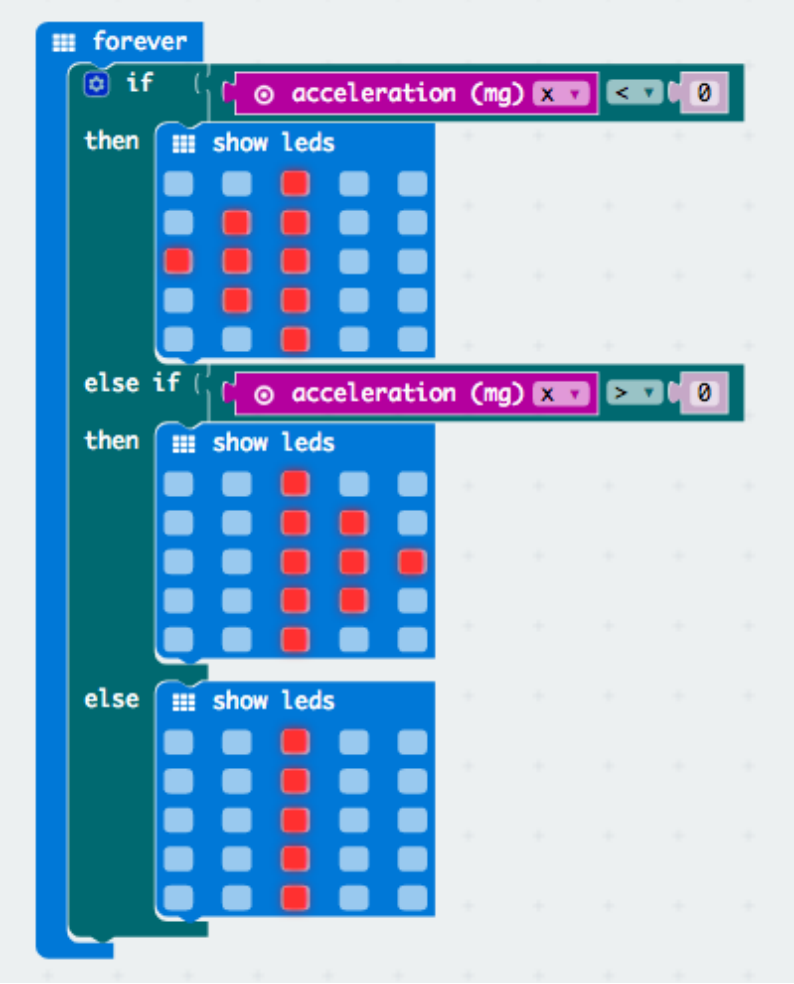


Figure 6.15: The block configuration for this particular exercise example

This exercise turns the micro:bit into a level by utilizing the a celerometer on the micro:bit. If the value of the accelerometer goes below 0 or above 0, feedback is given to user as to how to tilt the micro:bit in order to reach exactly 0 which is a level state.

6.1.2 Analog Workshop Example

This chapter contains all information necessary to host an analog workshop, rules of the game, materials needed, challenges and a plan on how to execute the workshop. This workshop is intended as an introduction to basic programming concepts for students without previous programming experience.

6.1.2.1 Rules

The rules are as follows:

- Each group will get a board and a set of cards
- Instructions are put after each other on the board
- The instruction are read from top to bottom, literally interpreted as a computer would have executed them
- Only use the instructions once
- No need to use all instructions
- The goal is to guide a friend through the challenge and reach last tile via instructions
- It is not allowed to step outside of the tiles, as this is considered as a violation

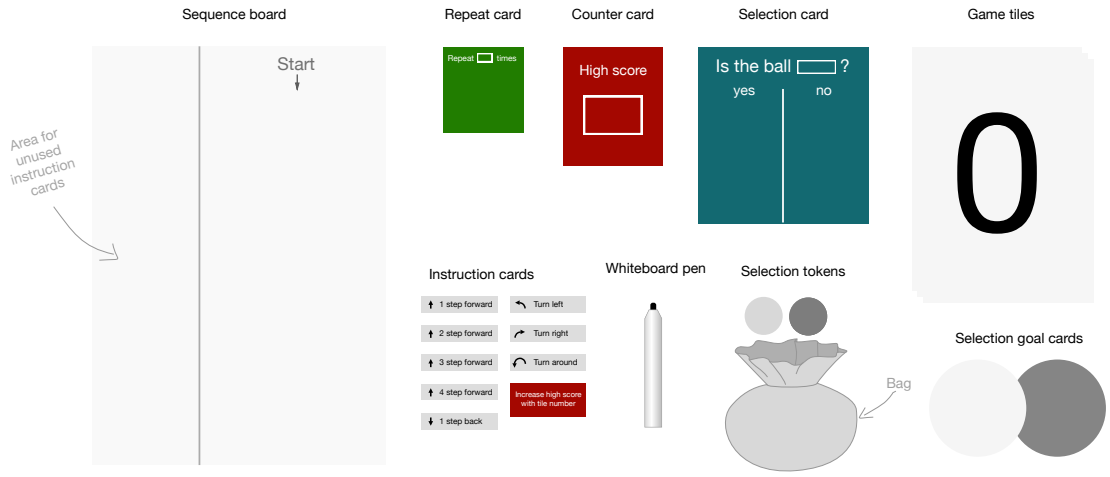


Figure 6.16: All material needed for both workshops

6.1.2.2 General Preparations

Create game tiles to create challenges, preferably numbered from zero to ten. A good way to keep them reusable is to laminate the tiles if possible. Plan which courses should be used, making own courses is really easy, and design for an appropriate difficulty level and also allow for the challenges used to be modified to give them longer playability. It is important to have at least one solution available to give hints if students get stuck.

Create instructions cards so there is enough for every group participating in the workshop, as seen in figure 6.16. These need to be planned together with the challenges used in order to make them solvable yet not trivial. These can be color coded to match the blocks in the MakeCode editor for the micro:bit. Provide boards or something similar so there is a surface to place the instructions and cards on. Cards representing coding blocks for repeat, selection and counter needs to be made as well. These work as mini boards that can be placed in a sequence on the board as well as hold instructions themselves.

6.1.2.3 First Workshop

Learning goals for this workshop:

- Become familiar with sequencing
- To be clear when providing a computer with instructions
- Learn how to recognize patterns in the code to reuse segments of instructions using a repeat card

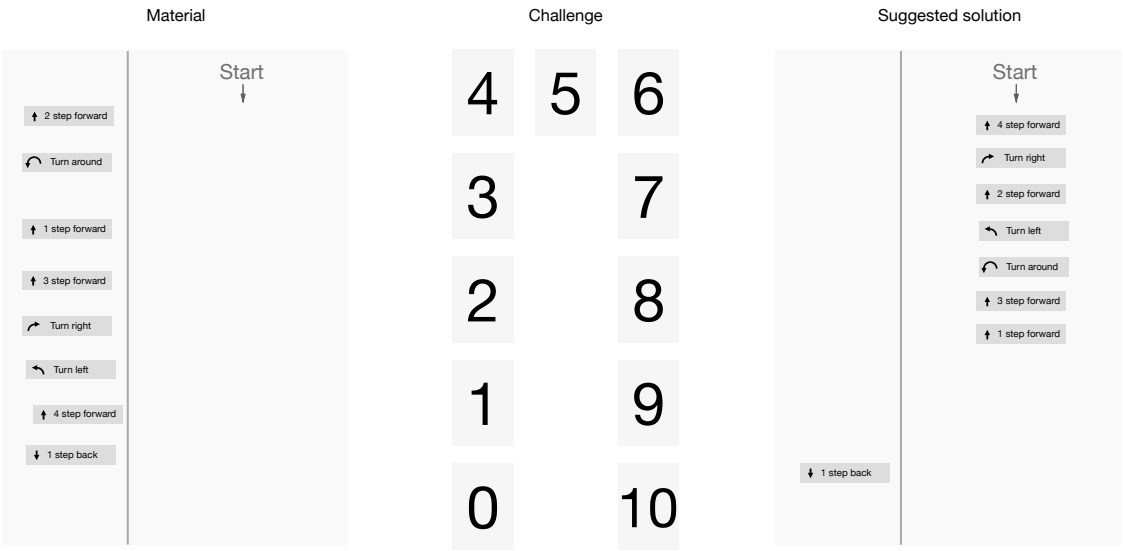


Figure 6.17: Material needed, layout and possible solution for the first challenge

Workshop plan:

- Divide students in groups 2-5
- Show the instruction board and explain the rules
- Demonstrate how to control a person via instructions
  - Make a sequence of any kind
  - Reverse the sequence to show that the order matters
- Let students try to solve the first challenge, figure 6.17, and let them present their result
  - When they have presented their results, remove the "1 step forward" instruction from their inventory(figure 6.18), then you have two choices:
    - Either solve the new problem together like co-coding
    - Give the groups additional time to solve separately

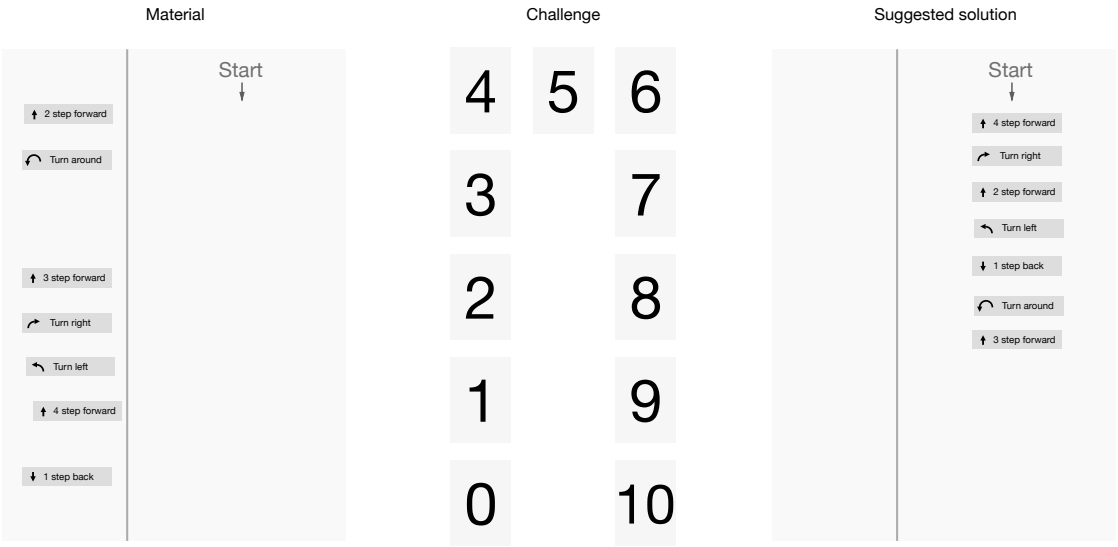


Figure 6.18: Modified version of the first challenge

Next, present the second challenge, figure 6.19

- Try to complete it together and realize that it is impossible (intentional)
- Add the repeat card and give an example of how it can be used
- Let the students try to solve the challenge with the repeat card repeat card

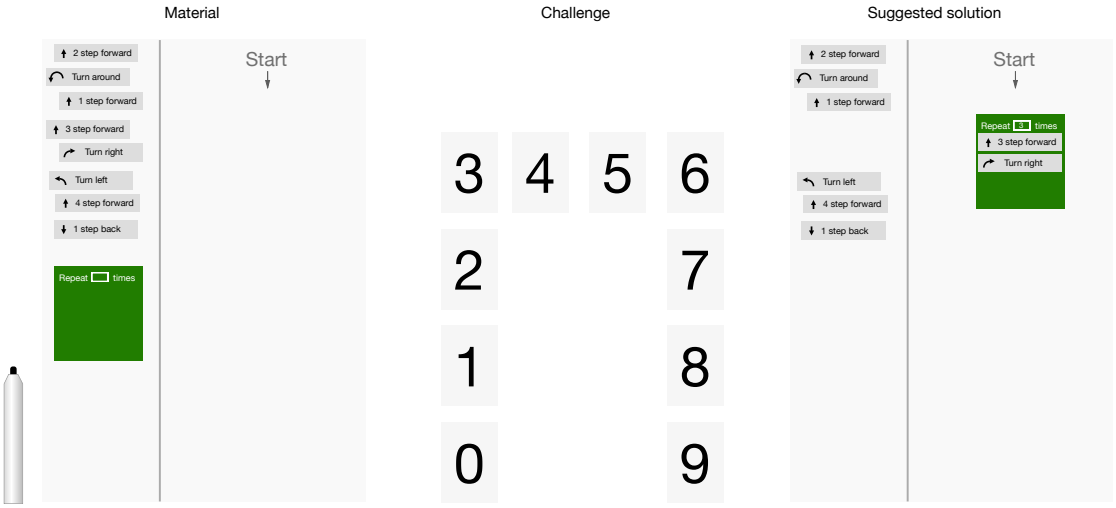


Figure 6.19: Material needed, layout and possible solution for the second challenge

Lastly, present a third challenge, figure 6.20

- Continuation on the exercises including the repeat card
- Let the students try to solve the challenge with the repeat card

Figure 6.20: Material needed, layout and possible solution for the third challenge

6.1.2.4 Second Workshop

Learning goals for this workshop:

- Understand the benefit of using the selection card
- Become familiar with the counter card, later identified as a variable
- Debugging, trying different code segments to solve problems

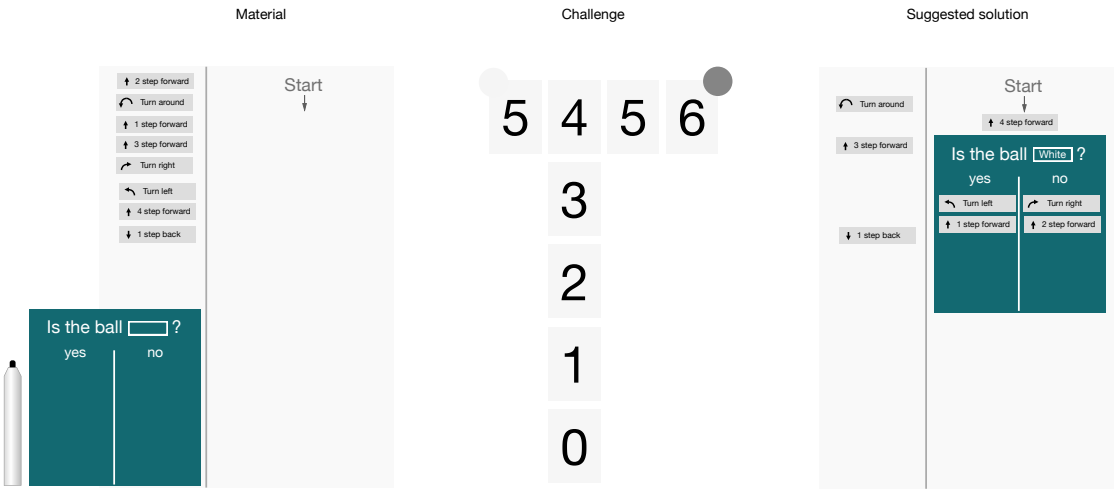


Figure 6.21: Material needed, layout and possible solution for the first challenge

Workshop plan:

- Divide students in groups 2-5
- Short repetition on the rules and repeat cards
- Introduce a new card, the selection card, and explain how it can be used
- Let students try to solve the first challenge, figure 6.21, and let them present their result
  - Increase the difficulty each time they solve a challenge, choose from two different extra challenges, see 6.22 and 6.23

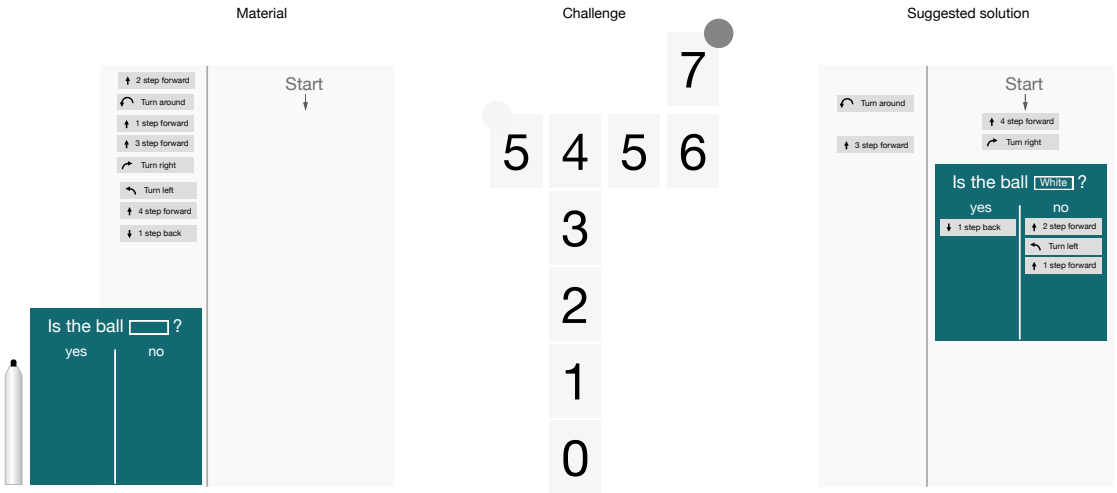


Figure 6.22: Material needed, layout and possible solution for an intermediate challenge

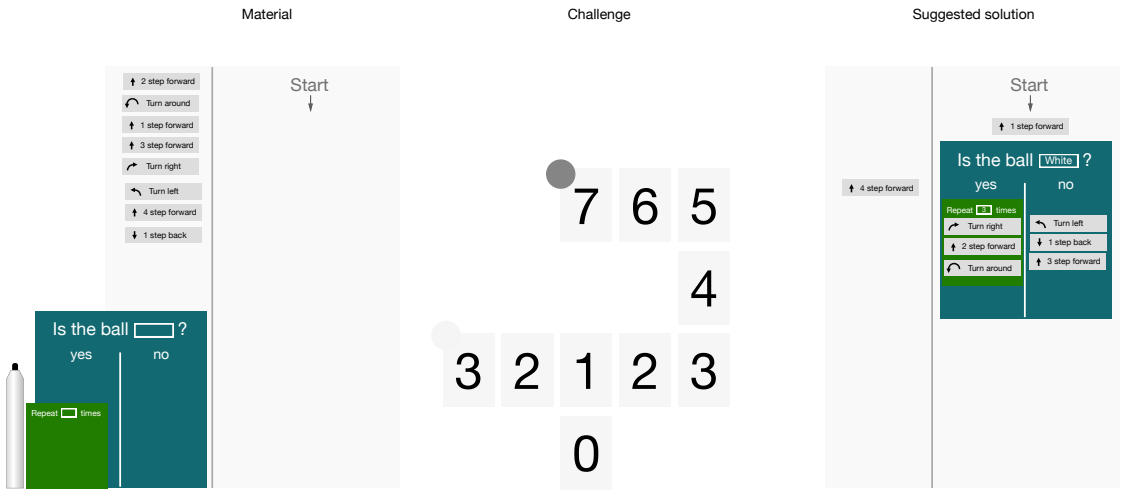


Figure 6.23: Material needed, layout and possible solution for a hard challenge

- Next introduce the counter card and show how it works together with the add score instructions
- Present the straight challenge, let students try to get as much points as possible, figure 6.24.
  - The counter card can essentially be used on any challenge to gather most points
- Next add the repeat card into the mix and let the students try and get more points, figure 6.25.

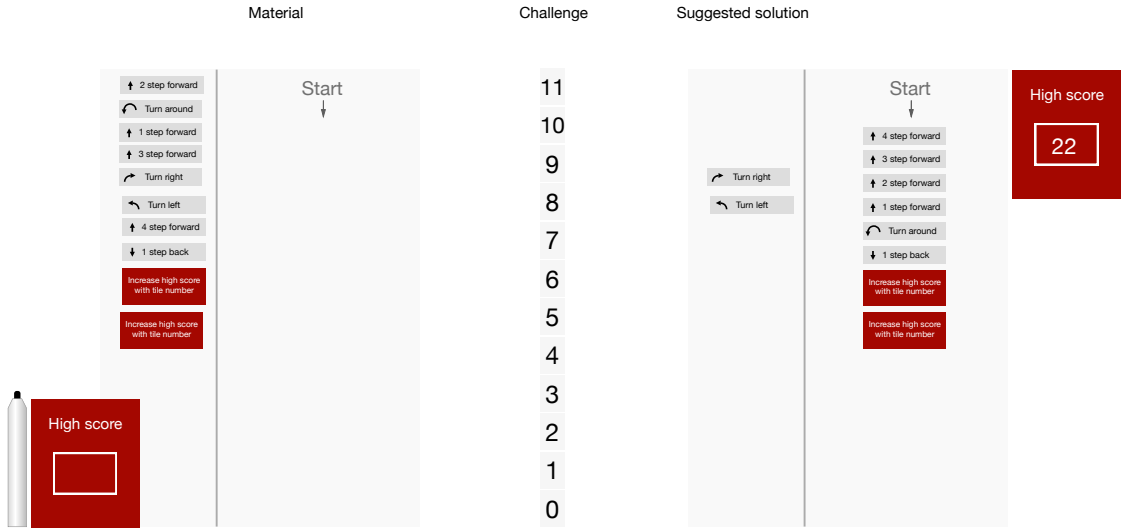


Figure 6.24: Material needed, layout and possible solution for the score challenge

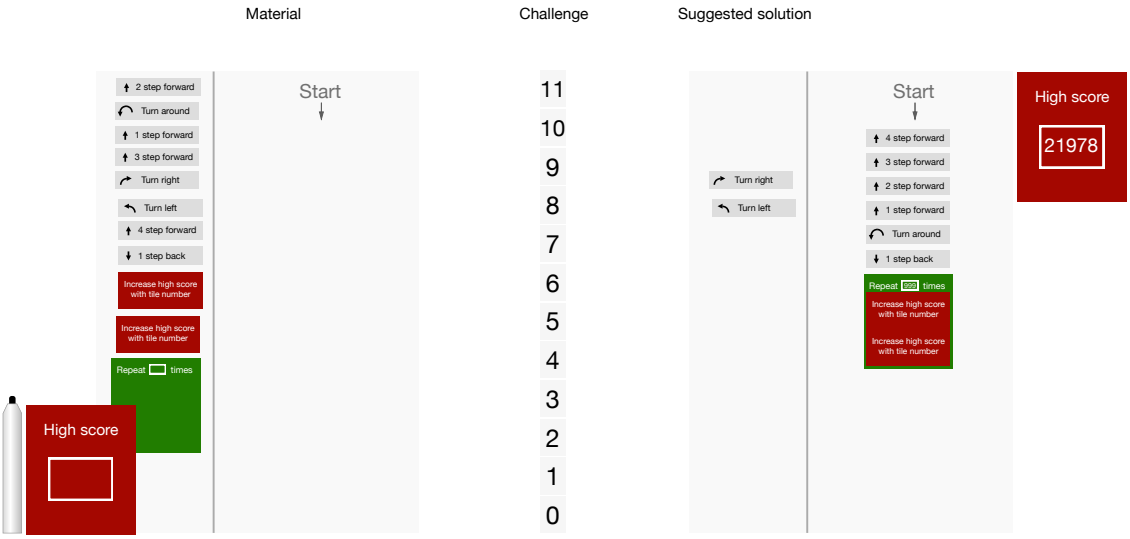


Figure 6.25: Score challange with an added repeat card

6.1.3 micro:bit Workshop Example

This chapter provides an example of two sequential micro:bit workshops, intended to be one hour each and for a class of approximately 25 students. The students are expected to have some prior knowledge of programming, either through the Analog Workshop example presented in chapter 6.1.2 or through a development platform comparable to Scratch.

6.1.3.1 General Preparations

In these workshops students are to work in pairs, hence one laptop, micro:bit, USB-cable and battery pack is required for each student pair. The battery packs are useful

as they allow students to use the micro:bit standalone from the computers. It can be useful to prepare the computers' web-browsers by activating the setting commonly known as: "Ask where to save each file before downloading". This will allow students to save the programs straight to the connected micro:bits, as it otherwise often becomes confusing where the downloaded files are saved on the computers.

It is possible to run the workshops with mobile devices, such as iPads instead of laptops, however this will take more time and one should, in that case, regard the technical pitfalls described in chapter 6.3.3.3.

These micro:bit workshops have been designed to be a continuation to the Analog Workshop Example already mentioned, as some concepts introduced here will be possible to relate back to exercises done in the Analog Workshop. Hence it is recommended to have done the Analog Workshop first, but it is not a requirement.

As this workshop makes use of the teaching approach Co-coding described in 6.2, it is a good idea to familiarize oneself with this approach. One should ensure that the classroom has a reliable internet connection and that there is a projector available with adapters compatible with the teachers computer.

6.1.3.2 First Workshop

The focus of this workshop is to introduce the students to the micro:bit editor and how to perform the procedure of moving programs from the computer to the micro:bit. It is recommended to run this workshop with computers, as it is easier and less time consuming than working with hand held devices such as iPads.

Start out by giving a brief introduction to the MakeCode micro:bit editor, showing it on the projector. Describing the different user interface elements, such as the folders of code blocks, the area where the code is built and the simulator. More information about the editor can be found in chapter 2.3.1. It is a good idea to write down the URL-address on a whiteboard. By the time this thesis was written the URL was: makecode.microbit.org, this might however change over time. The students may now visit this URL on their own computers.

Next it is time to hand out one micro:bit, USB cable and battery pack to each student pair. When all students have gotten their hardware and managed to get the editor up and running, it is time to get everyone's attention. Emphasise the importance of paying close attention to the creation of this first program, as they will need to know these steps and it will save a lot of time and confusion.

In case there are no computers available for this workshop and it has to be entirely run with hand held devices such as iPads, ensure that the students direct their undivided attention to a complete demonstration of how to pair and flash code to the micro:bit, before even handing out the hardware. More information about pairing the micro:bit to hand held devices can be found in chapter 6.3.3.3.

Now co-code a name badge exercise as described in 6.1.1.2 and ask the students to imitate what is being done on the projector. Demonstrate how it can be tested in the simulator and press the download button in the editor to save the .hex-file to the computer. If the browsers have been setup to prompt where to save each file, it is now possible to connect the micro:bit via USB, and choose to save the .hex-file to the device called MICROBIT that appears as a thumb drive among the computer devices.

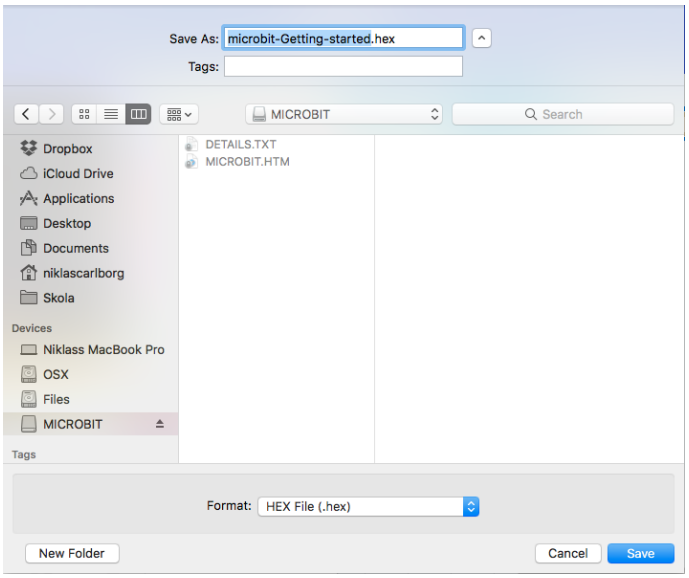


Figure 6.26: With the browser set to ask where to save each downloaded file, it becomes easier to save the programs straight to the micro:bit without having to locate them and manually move them with a file manager

At this point some time will probably be required to walk around and help students troubleshoot. Students who have gotten help or managed to complete the exercise without help, can be encouraged to help others.

Next co-code a step counter as described in chapter 6.1.1.6. While co-coding this it can be a good idea to keep a dialogue with the students about how a step counter works and that it needs to remember how many steps have been taken. If the students previously have done the Analog Workshop, this is a good opportunity to refer back to the exercise with the high score counter. As this situation is quite similar to that. Here the variable blocks can be introduced as something that works just like the high score counter from that exercise. After the students have managed to create their own step counters, they may be encouraged to connect the battery packs to the micro:bit and try attaching their step counters to their feet and have some time to walk around with them. This can be allowed to take some time as it often is appreciated.

In case the time for the workshop is running out at this point it is recommended to end here with a positive atmosphere, rather than trying to force in another exercise. But if there is still 15 minutes left, the coin toss exercise can be co-coded as well. The

blocks for this can be seen in chapter 6.1.1.3. In case the students previously have done the Analog workshop, this is a good opportunity to refer back to the exercise where they had to pick a random ball to determine where to go in the game. As this resembles the situation of programming a random coin toss on the micro:bit.

6.1.3.3 Second Workshop

The focus of this workshop is to allow the students to complete exercises in a more problem solving manner, compared to the previous workshop where they simply imitated the teacher. As this is the second workshop and still very early in their programming education, this workshop should begin with a recapturing of what was done in the previous workshop. Briefly go through how to upload code to the micro:bit, editor navigation and putting blocks together. Proceed to co-code a Dice, as seen in figure 6.6, together with the students. As this is a completely new program but something they all are familiar with, a six-sided dice, discuss the idea of how such a program would work and begin building it together. This is also a great opportunity to get a sense of where they are at with their computational thinking, be responsive to any signs of confusion and clear them out. Techniques learned in this exercise will aid them in the next one, where they are encouraged to build something completely new.

The main exercise of the second workshop is the Rock Paper Scissors(figure 6.7), a game most people are familiar with. Begin by having an open discussion in class on how the game works in reality by writing it down in detail, as the computer will need clear instructions to work as desired. Then try to think as a computer with the knowledge the students have gotten thus far, bringing up the Dice example for inspiration and reference. Without showing any blocks or code, let the students try to program Rock Paper Scissors themselves without any further instructions. After approximately fifteen minutes provide all the blocks (if needed) that is required to complete the program, although unconnected and scrambled to give students a nudge in the right direction as seen in figure 6.27. Be sure to encourage those that have pursued a different kind of solution as that could be of great interest for other students.

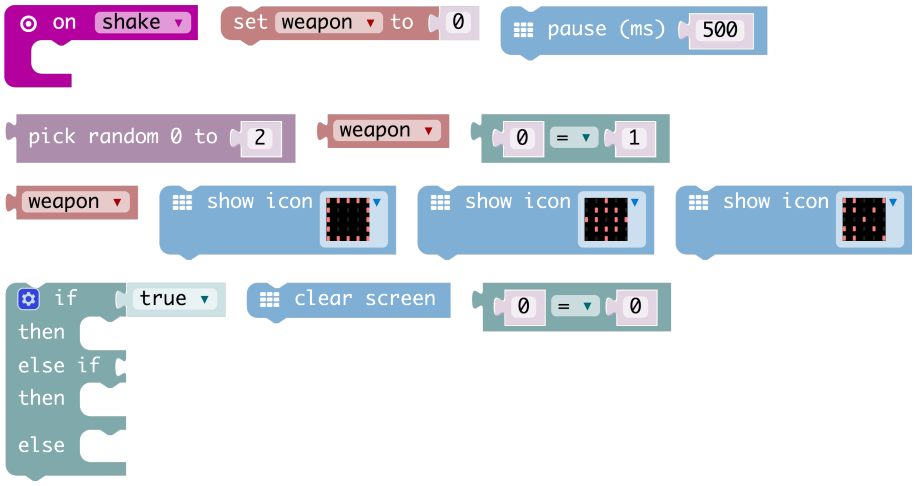


Figure 6.27: Blocks needed for the Rock Paper Scissors game, scrambled.

In the last fifteen minutes of the class be sure to co-code the whole Rock Paper Scissors program together with the students, explaining block choices and the steps taken to arrive at a working solution. If there were students completing the exercise with a different solution, let them present their choices and steps taken as well.

6.2 Co-coding Teaching Approach

Co-coding relates to the activity of having the teacher stand in front of the class, screen sharing their computer screen on a projector and solve programming exercises in a dialogue with the class. This form is a useful hybrid between pure presentations and having students solve exercises on their own. Presentations were considered to be good for introducing new knowledge, but it was undesirable to put the students in a rather passive seat. On the contrary, working individually with exercises, was considered to be more active but not ideal for introducing novel information. Co-coding hence evolved as a middle path between these two approaches, as illustrated in figure 6.28. Our experience was that it was a useful method as it allows the teacher to probe students current skill level with questions and rate the discussions in the class and adapt the level of guidance. This method was to some degree inspired by Hattie and Donoghue(22) mentioning of teaching learning strategies in context rather than separately. Co-coding can be an effective method to teach new concepts, like the need for variables for instance, within the context of an exercise, rather than as a separate presentation. Co-coding should be practiced often as it provides an including activity for the whole class, prepares students by carefully giving them new tools to work with and also lets the teacher get an overview of the general understanding towards the subject in the class.

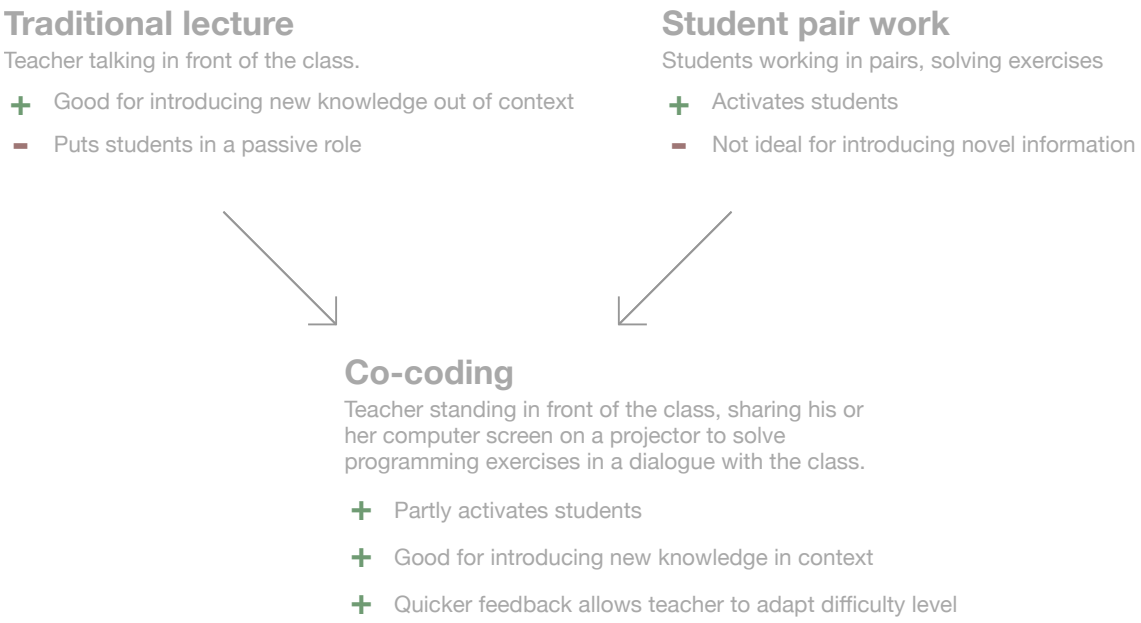


Figure 6.28: Co-coding as a combination of lectures and student work

6.3 Guidelines

The presented workshop examples are based on insights that have been derived throughout the project. Some of these major insights are here presented in the form of guidelines so that Swedish teachers who would want to use them can create their own micro:bit workshops, rather than use the provided workshop examples. Three of these guidelines were aggregated from multiple insights from along the project timeline, whereas eight others were single insights that were found useful to share. The topics for the three aggregated insights are: basic toolbox, terminology and technical pitfalls. The single insights are presented as other considerations.

6.3.1 Basic Toolbox

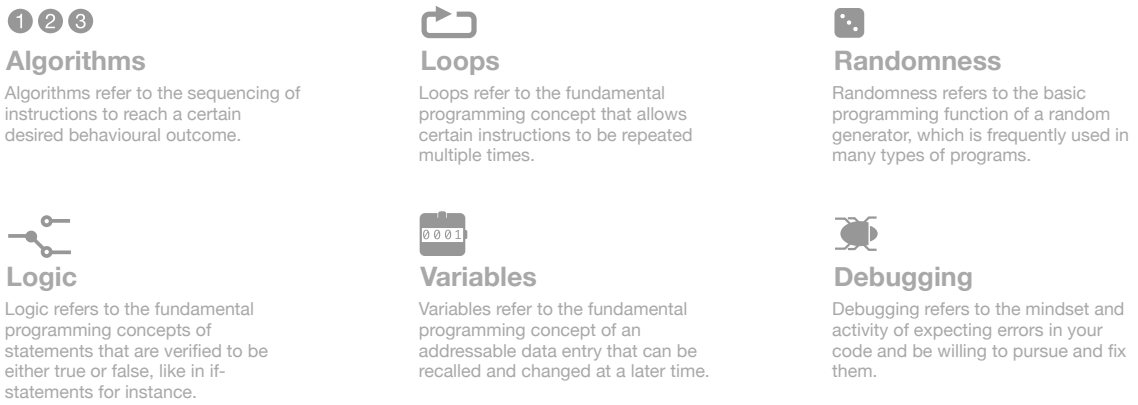


Figure 6.29: An overview illustrating the basic toolbox

Basic toolbox relates to a set of fundamental programming concepts that were found useful for students to have been introduced to, prior to working with programming exercises. The programming concepts that were found useful for this were: algorithms, loops, randomness, logic, variables and debugging. It was found beneficial to introduce and practice these concepts in parallel to each other, rather than to work with each one of them separately, in series. This due to the perceived difficulty to create interesting exercises based on only single programming concept. What was seen as interesting exercises, were combinations of multiple programming concepts.

6.3.1.1 Algorithms

Algorithms refer to the sequencing of instructions to reach a certain desired behavioural outcome. It has been seen that students benefit from having been introduced to this concepts prior to start working with programming the micro:bit. Algorithms are also mentioned in the Swedish government's policy changes regarding the documents that control Swedish primary and secondary school curriculum(41). Introducing students to this can be done through many possible approaches. The way it was introduced in this project was through having the students control each other with instruction notes through a game field. This game later came to be called analog workshops.

6.3.1.2 Loops

Loops refer to the fundamental programming concept that allows certain instructions to be repeated multiple times. It has been seen that students benefit from having been introduced to this concepts prior to start working with programming the micro:bit. Combining loops with other tools from the basic toolbox was seen as necessary to create stimulating and useful exercises with the micro:bit. It was found useful to teach loops in the context context rather than simply as an abstract concept.

6.3.1.3 Randomness

Randomness refers to the basic programming function of a random generator, which is frequently used in many types of programs. It has been seen that students benefit from having been introduced to this concepts prior to start working with programming the micro:bit. Having knowledge about how to use the random function when working with micro:bit proved beneficial when starting to learn programming. Since many of the beginner projects use elements of randomness as a function in their code, such as roll a dice or rock-paper-scissors, being familiar with it became important to complete those programs.

6.3.1.4 Logic

Logic refers to the fundamental programming concepts of logical statements that are verified to be either true or false by a program. This for instance is a central part of if-statements. It has been seen that students benefit from having been introduced to this concept prior to start working with programming the micro:bit. The way logic

was introduced to students in this project was through having the students control each other with instructions through a game field. Some of these games required the students to create alternative instructions depending on the outcome from a random event. More details about the implementation of logic in this game can be found under the section analog workshops.

6.3.1.5 Variables

One truly fundamental concept of programming is the one of variables, as without variables and other data structures there is no way for a computer to store information. This information can be of different types: values, names, and might prove difficult to grasp at first for students. Nevertheless it has been seen that students benefit from having been introduced to this concept prior to start working with programming the micro:bit. Many of the beginner projects involve the usage of variables, therefore it has an obvious place in a basic toolbox for students. The way variables were introduced to students in this project was through the metaphor of having a high score counter. As students moved through the game later known as analog workshops, they got to add the value of the tile they were standing on to their high score counter. More details about the implementation of variables in this game can be found under the section analog workshops.

6.3.1.6 Debugging

Debugging refers to the mindset and activity of expecting errors in your code and be willing to pursue and fix them. Programming without ever encountering any errors, or bugs, is highly unlikely. Fixing errors can be time consuming but is an essential skill for programmers to learn. As students have been seen able to enter states of indifference or dejection when faced with errors, it is considered useful for students to have been introduced to this approach prior to start working with the micro:bit. Regarding block programming on the micro:bit it rarely becomes a syntax or coding error due to the nature of blocks. The errors are more likely logical errors. Solving these errors is considered to be an important part of deepening the understanding for programming. There are some steps one can practice when confronted with a bug. Firstly it is a good idea to try and predict what the program should do, and step by step execute the code manually out loud to try and see where the error is. This way the problem often becomes apparent rather quickly. In the analog workshops game in this project debugging was not designed into the exercises per se, and it was verified that debugging still was practiced simply by working with the exercises. The role of the facilitator or teacher simply was to encourage debugging when the inevitable errors appeared.

6.3.2 Terminology

Terminology is referring to the words used when teaching programming. As there are many new words and concepts that might be intimidating for students at first, it is useful to initially avoid using words such as variables, but rather attempt to convey these through words and concepts that already are familiar to the student.

For example the concept of a variable can be described as a high score counter in a game, something that the student might already be familiar with. This does not say that variables are precisely high score counters, but it is a way of conveying the intuitive concept without introducing any new potentially scary words. It can still be encouraged that the students learn the correct terms for concepts, but it is a matter of easing them into it. From a teacher's perspective it can be hard to be aware of when one uses programming terms, therefore it is advised to pay close attention to the language one uses so that no new words are introduced without proper introduction first, preferably linked to previous knowledge. Some qualitative signs have indicated that it might be easier for a student to do something first and then afterwards learn the proper name for it, rather than first being introduced to a new word before learning what it is about.

6.3.3 Technical Pitfalls

Technical pitfalls refer to practical issues that have been identified to risk obstruct or fail the execution of a workshop. As it has been seen that students can enter states of indifference or dejection if these issues take up too much time, it is suggested to take measures and attempt to prevent them from arising in the first place. Three specific issues that have been observed are relating to: app-store passwords, internet connection and pairing mode bugs.

6.3.3.1 App-Store Passwords

In the context of running a micro:bit workshop with iPads, the students are required to download the micro:bit app from the app-store. This requires an app-store account and as some schools prefer to control what apps are being installed on their iPads, some schools protect their accounts with passwords. It is therefore recommended to obtain these passwords well in advance and work out a good way for these apps to be downloaded in class, or possibly even downloading the micro:bit app to each individual device in advance.

6.3.3.2 Internet Connection

Unreliable internet connection was seen as another frequent source of frustration. As the micro:bit editor is run through the web browser reliable internet connection is required throughout the workshops. Technically the Microsoft MakeCode editor for micro:bit can be used offline as the application gets cached locally although an online compilation has to be made first. In cases where internet is slow or the connection dropped occasionally, students will get frustrated and precious learning time will be spent troubleshooting internet connections instead. In those cases where workshops rely on online material, such as instruction videos, the internet bandwidth plays an even more noticeable role, as video can be rather bandwidth heavy.

6.3.3.3 Pairing Mode Bugs

Lastly there were some issues identified regarding the pairing of micro:bits to iPads. Firstly there is a certain procedure that is required to initially pair a micro:bit to an iPad. This procedure can be rather tricky at first, as it requires the student to press three buttons in a certain order and remember a series of six numbers shown in a rapid sequence. Students grasp the pairing process the fastest by being shown the full procedure and then try it themselves. Having students imitate this procedure in real time is not recommended, as it can lead to disorder in the class.

Another issue one might encounter is a bug relating to sending a program (flashing) from a mobile device to a paired micro:bit. This process requires the micro:bit to be put into pairing mode again, despite that this is not mentioned in the documentation. As this bug is not documented it can be hard to solve and can hinder an entire workshop from progressing. Lastly there have been a few rare occasions where micro:bits were impossible to put into pairing mode. This bug is resolved by simply having a computer nearby and flash any type of program from the computer to that micro:bit via USB-cable. This way the micro:bit gets reset and can be paired with an iPad again.

6.3.4 Other Considerations

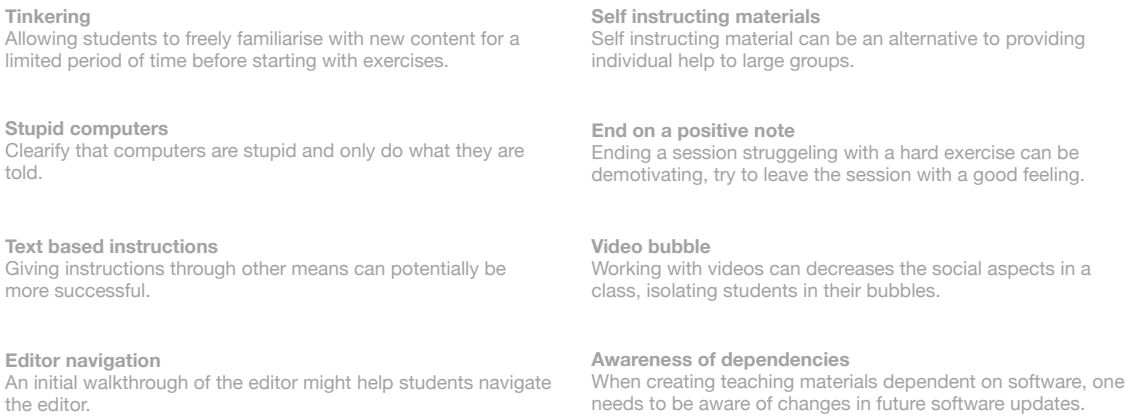


Figure 6.30: An overview of other considerations

Furthermore there were eight single insights from workshop interventions that were found useful. Since these were not based on aggregated observations however, they are here presented as slightly less significant considerations. The topics related to these are: tinkering, self instructing materials, stupid computers, end on a positive note, text based instructions, video bubble, editor navigation and awareness of dependencies.

6.3.4.1 Tinkering

Tinkering is allowing students to freely familiarise with new content before starting to work with exercises. This explorative approach without any goals or objectives

was seen as a way for students to get outlet for any curiosity that might arise as they are introduced to new technology, platforms or concepts. To prevent frustration or confusion to arise, it is recommended to keep this initial tinkering short in time.

6.3.4.2 Stupid Computers

Stupid computers refers to making it clear to students that computers are simply following instructions and should not be considered as intelligent per se. In some cases we have seen students who expect that computers are smart just because they are computers. Clarifying this initially can potentially prevent some of these misconceptions.

6.3.4.3 Text Based Instructions

Purely text based instructions might not be the optimal way for conveying exercises. Giving instructions through other means can potentially be more successful. Students mostly ignore printed papers with instructions that have been handed out. Conversations with students showed that they found written instructions incomprehensible, and prefers facilitators to answer their questions. Other means of giving instructions could also be through video.

6.3.4.4 Editor Navigation

Editor navigation refers to the need to understand how a certain programming editor works and how to navigate it in order to use it. It is easily overseen and can be considered trivial by someone familiar with it, nevertheless is it important for a first time user.

6.3.4.5 Self Instructing Materials

Self instructing materials such as instruction videos, have been seen as potentially useful in cases where student group sizes exceed the number of students that the facilitators are able to provide help to. Even if the class size is manageable self-instructing material allows students to work at their own pace and given that the students are ready for the exercises it can work as an offload for the teacher, which can focus their help where most needed.

6.3.4.6 End on a Positive Note

End on a positive note refers to concluding workshop sessions with an exercise that leaves the students in a positive state of mind, rather than leaving them confused or frustrated. Struggling with exercises might be important for the development of grit, the harder exercises should be placed in the middle of a session and allocate the end for easier ones that allow students to feel successful. From a facilitator point of view this means planning the timing of the session well, and never try to cram any exercise in the very last minute, just because you want to convey something that might not really have gotten across. Energy levels of the class is usually rather low

in the end, and trying to force last minute teachings in here, seems to possibly make more harm than good.

6.3.4.7 Video Bubble

When using instruction videos as teaching material for an entire class, one ought to be aware of the potentially negative effects this might have on social aspects of the group. A classroom full of students watching different videos on their computers will get rather noisy, headphones are recommended. This however also has the effect of isolating students into their own video bubble. This might be positive for some, in terms of concentration, but it also removes many of the interpersonal social interaction that can be positive in a group. This way of providing video material is probably better suited for homework of some sort.

6.3.4.8 Awareness of Dependencies

When creating teaching materials dependent on software, one needs to be aware that it can be changed in future software updates. For instance if one creates a set of instructions on how to navigate an editor that in detail refers to certain buttons, the names of these buttons may well be changed in future software updates. To avoid the risk of confusing students, it is therefore suggested to continuously verify that the teaching material with dependencies is up to date with the current state of the software. This was discovered as teaching materials, that intentionally had been colour coded to match the micro:bit editor, turned out to no longer match the colours of the editor at a workshop.

6.4 Scope of Autonomy Model

Here the scope of autonomy notion will firstly be described, accompanied by five levels of autonomy identified for working with the micro:bit. These two parts make up the scope of autonomy model. This model has been suggested as a result from the empirical qualitative research performed throughout this thesis project. The model attempts to explain observed behaviours and phenomena regarding Swedish primary school students encounter with programming the micro:bit. The model is intended to be used as a tool when creating exercises, to help teachers bring awareness to the amount of choice expected of students within exercises.

6.4.1 Scope of Autonomy

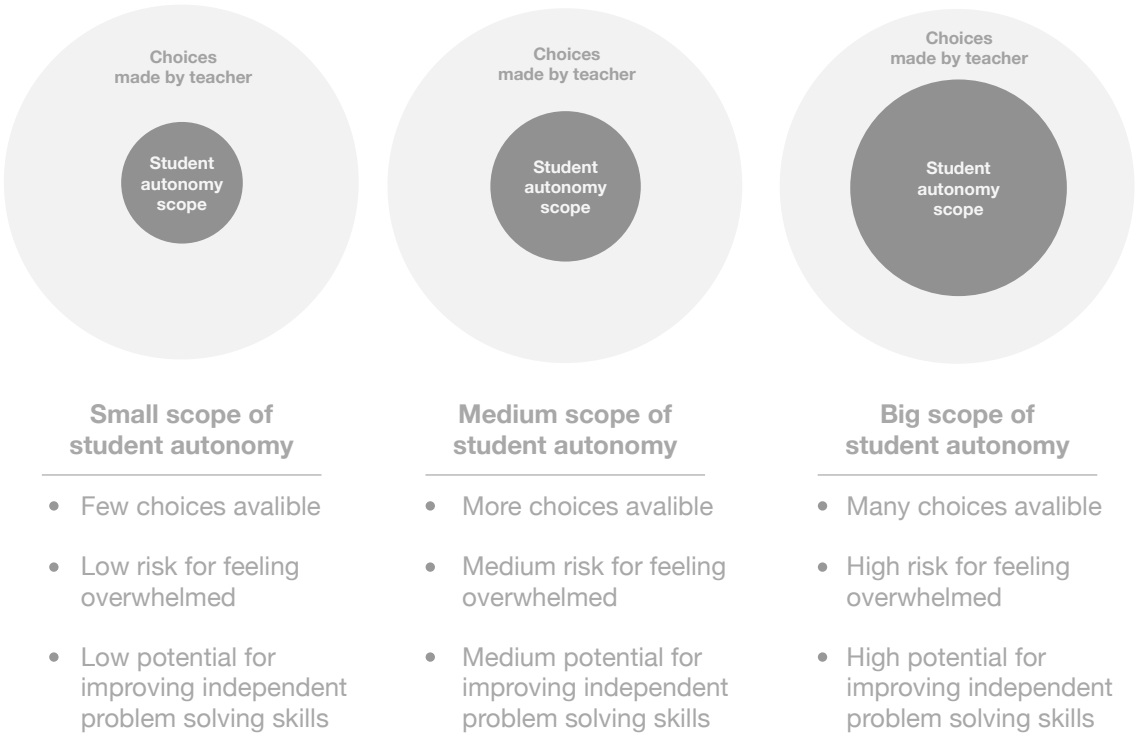


Figure 6.31: Different scopes of autonomy

This is a model to illustrate and bring awareness to the distribution of autonomy between students and teachers in relation to single given micro:bit exercise. The model is based on the premise that completing an exercise involves making a set of choices. The dark area in the center of the model represents the choices made available to the student, this is called the students scope of autonomy. The gray area surrounding it represents the choices made by the teacher. A larger student scope of autonomy hence implies fewer choices to be made by the teacher.

The model suggests that the larger the scope of autonomy becomes, the higher the student runs a risk of feeling overwhelmed.

The model also suggests that the larger the scope of autonomy becomes, the higher the potential is for the student to improve their independent problem solving skills.

Hence there is according to the model a balancing act in the creation of an exercise. So that it provides the students with enough choices to develop their independent problem solving skills, yet without exposing them to too many choices so that they feel overwhelmed.

The model does not make any claims on how to determine what the appropriate level of autonomy is for any student.

6.4.2 Micro:bit Levels of Autonomy

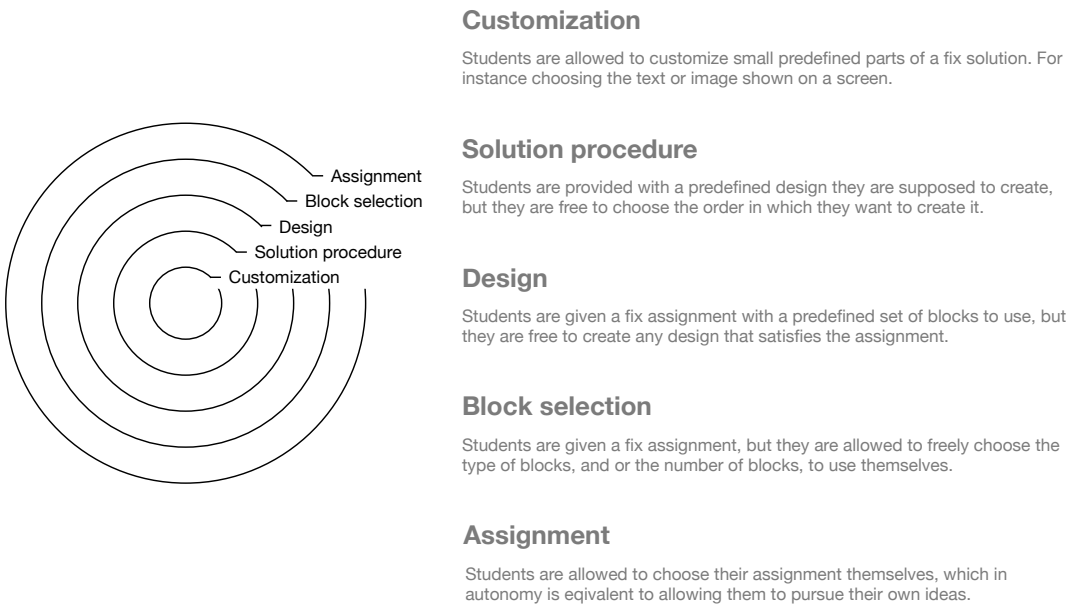


Figure 6.32: Each circle represent a level of autonomy

Five levels of autonomy were identified for working with the micro:bit. These are presented in a radial fashion to be compatible with the scope of autonomy model. Any micro:bit exercise is supposed to be mappable as a scope of autonomy disc onto this model. The more of these levels that are encompassed by an exercise the bigger scope of autonomy it has.

6.4.2.1 Customization

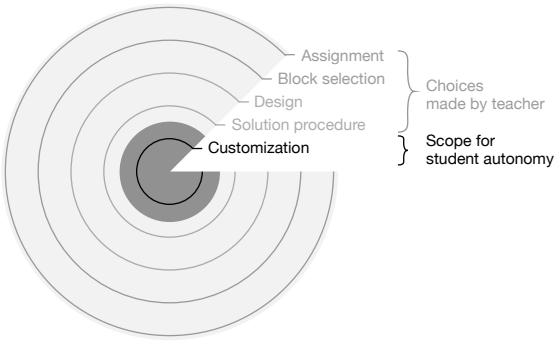


Figure 6.33: Scope set at Customization level

The first level of autonomy that was identified in relation to micro:bit exercises, was allowing students to make smaller customization to a predefined design. In the case of a simple "hello world" program, this could mean allowing the student to customize the text string to something else than "hello world". Hence a customization is not something that alters the behaviour of a design, but rather allows the student

to locally modify specific point of interest that have been selected by the person designing the exercise. From figure 6.33 it is possible to see that a majority of the choices that have to be made regarding the exercise still has to be made by the teachers when an exercise has this scope of autonomy.

6.4.2.2 Solution Procedure

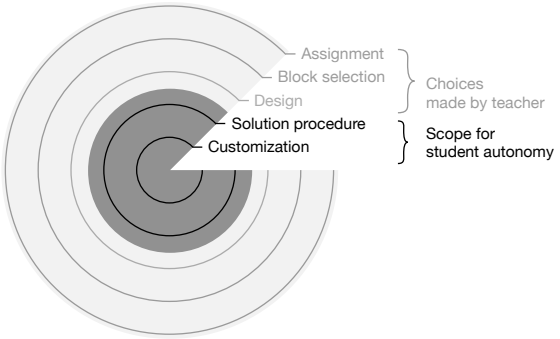


Figure 6.34: Scope set at Solution procedure level

The next level of autonomy that was identified is the one relating to the solution procedure of an exercise. This level of autonomy relates to what subparts of a solution to tackle in what order. When this level of autonomy lies within the student's scope, the student is free to choose the order in which to create the solution. When the solution procedure does not lie within the scope of the students autonomy, the students are asked to follow a stepwise procedure instructed by the teacher. In the case of creating an animation, this could relate to the difference in starting with drawing the desired animation and then figuring out the best timing between frames, or doing it the other way around. This way there is more freedom for the student to make choice about the way they solve an exercise but the target design is still chosen by the teacher, as illustrated by figure 6.34.

6.4.2.3 Design

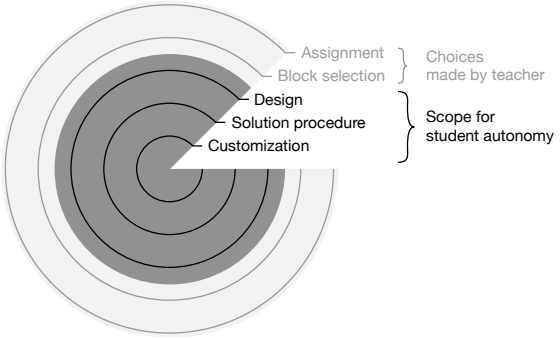


Figure 6.35: Scope set at Design level

The third level that was identified, is concerned with the design that a student makes to complete an exercise. This is a rather interesting level of autonomy, as setting the

students scope of autonomy to encompass this level means that the teacher no longer knows what the final design will look like, as it is up to the student. In contrast to a scope of autonomy that only encompasses the level of customisation, a scope that encompasses the level of design allows for completely new design solutions to an exercise, and not only the modification of predetermined placeholders. The student is however still restricted by the teachers choice of blocks to be used with this scope of autonomy, as illustrated by figure 6.35.

6.4.2.4 Block Selection

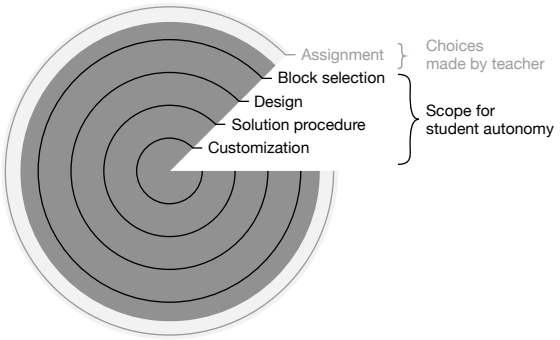


Figure 6.36: Scope set at Block selection level

The fourth level of autonomy that was identified is related to block selection. Blocks are the building pieces that are used to create a design. When this level is not encompassed by the students scope of autonomy in an exercise, it means that the teacher has predetermined what blocks the student should use to create his or her design. This level of autonomy has two parts. The first of which relates to the type of blocks and the second one relates to the quantity of blocks. For instance the teacher can give an exercise where the students are asked to create an animation on the micro:bit using any number of blocks of the "loops" and "show LED" variety. This way the types of blocks are chosen by the teacher but the student is free to choose the number of blocks. Another exercise could be to make a step counter using only four blocks in total. Here the teacher decides the number of blocks but their type are free to be chosen by the student. These two examples illustrate that an exercise can be created in ways where the students scope of autonomy only encompasses one of these two block selection levels. Likewise none of them can be encompassed, which means that the teacher decides exactly what blocks ought be used. And lastly when both of them are encompassed by the student's scope of autonomy, it means that the student is free to create a design out of any block type or quantity, as long as it satisfies the assignment. This is illustrated in figure 6.36.

6.4.2.5 Assignment

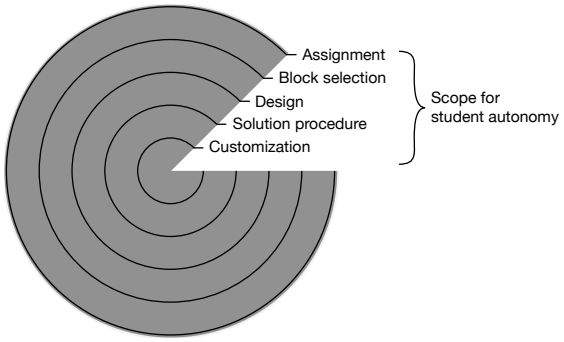


Figure 6.37: Scope set at Assignment level

Lastly an autonomy level was identified relating to the very assignment itself. This relates to decisions about the topic and aims of an exercise. In the case where this level is not encompassed by the students scope of autonomy, the teacher defines what the student ought to do in order to complete the exercise. When this level is encompassed by the students scope of autonomy however, students make the decisions about what the exercise is going to be about. These kind of exercises might initially only be associated with higher educational projects, it is however just as true for exercises where the teacher tells students to create whatever they want. As having to create your own assignment basically is the same as having to come up with an original project idea. For being able to handle this level of autonomy however, students are recommended to have reached a rather high level of experience and be comfortable with making various decisions, or they might run the risk of feeling overwhelmed. As illustrated by figure 6.37 this level of autonomy does not require the teacher to make any decisions.

7
Discussion

The following chapter will be discussing the process of the project, some reflections on the result as well as possible future work.

7.1 Reflection on Process

The process was intended to be an iterative design process with a few well defined iteration cycles. The way it turned out however was that the sharp boundaries between these well defined iteration cycles got rather blurred. There was still a smaller ideation phase associated with each intervention, as well as an evaluation afterwards. This was done through the writing of documents before each activity, stating why we wanted to do it and what kind of information we were looking for, followed by a script for the activity. Shortly after each activity these documents were expanded with a few notes on how it went and what we learnt. These documents later turned out to be more valuable than expected as the small insights were summed up and used to an aggregated result. Without these activity documents we would probably have forgotten many of the small insights along the way, and been left empty handed in the end. The number of workshops also greatly exceeded the initial planning. We planned to do 4 workshops and ended up doing 21. Not all of these had their own unique planning and evaluation however, so it would not be fair to say that we did 21 iteration cycles. Out of these 21 workshops 10 can be considered to be small but complete iteration cycles. In the plan we set aside individual timeslots for prototyping, this however turned out to get merged into the ideation and execution of activities. So the need for separate prototyping planning was not really necessary, as the whole process was prototyping.

Furthermore the difference between what was pure user research and what was pure design iterations, also got somewhat blurred. The initial idea was to only gather insights about users, before starting to create workshop materials. But it turned out that we needed to create a workshop in order to observe the users in it. Hence it became somewhat of a chicken or the egg dilemma, and we ended up doing a bit of both in parallel. Still the first workshop interventions were more of introductory nature and mostly focused on gather qualitative data about the users. Some tweaking of the content and execution of workshops still occurred but the focus was not on improving the workshop design but rather to observe the behaviour, needs and progress of the students in their first encounters with the micro:bit, and in some cases even programming.

The initial research question was: “How can teaching materials for digital literacy learning, based on micro:bits, be designed in order to accommodate the needs of teachers and students in the Swedish primary and secondary school?”. This was however changed after discussion with the faculty examiner, as it was considered to be more academically useful to answer what to consider, rather than simply developing one single suggestion on how to design something.

The writing of the planning report, and especially the researching of papers, took much more time than expected from when the first plan was created in the proposal. Having to write the planning report, with all its theory, this early, was however considered to be rather useful. As these theories indeed did help us get unstuck throughout the project and fed into many of the ideas. Regarding getting unstuck at times of despair, such as the final analysis phase, getting feedback from our supervisors was very helpful and we are thankful for all the fresh input we got from them.

At the beginning of the project there was only a suggestion about curriculum changes from the National Swedish Agency for Education. Hence there was still a bit of uncertainty at this point, to whether or not these changes would actually be taken into effect. It was not up until the 9th of March 2017, that this was confirmed by the Swedish government.

By the end of the research phase we wanted a good way to summarise the collected user data to look for deeper insights. Spontaneously we wanted to try making a journeymap and personas, in the hope that this would lead to a better understand for the users. To some extent it did help us, but looking back, the work put into it was not in relation to the perceived insight we got out of it. This might very well be due to our limited knowledge about how to put the method to use, or the fact that the data fed into it was solely made up of exit tickets.

The interview with an experienced workshop facilitator gave us some new insights and ideas and was beneficial to much of the following work. As the interview was undertaken previous to our collaborations with Västergårdsskolan, it provided us with ideas of including analog elements in the workshops. Looking back at the benefits from a single workshop it could be argued that we should have made more interviews in our research phase. Although we had talked to many teachers along the way, no official interviews took place.

Working with the same class at Västergårdsskolan over a couple of weeks time was perceived as a good way to work with the progression of students. Although we regret not doing it sooner since the lack of time available had us end it earlier than we wished for. In hindsight we might have been better off doing this segment earlier and over a longer time as we felt we really learned something each session. In contrast to only performing first-time workshops which easily felt forced and cramped due to the short time.

Using the affinity clustering method to summarize all the data from the project, turned out to be a non linear, chaotic, time consuming and at times frustrating activity. However it turned out to be fruitful and we have no regrets using it. In parallel with this process we created graphics to model our thoughts and ideas. These graphics later helped shape the scope of autonomy model and clarify our own ideas to some extent. Many of the graphics and mind maps created throughout the project were means for processing information and ways of thinking, rather than something intended to be used for presentation.

7.2 Reflection on Result

When creating an exercise for working with micro:bit there seems to be some importance in making a conscious decision regarding its scope of autonomy. So that the exercise matches the student's current level as good as possible, and provides them with an opportunity to improve their independent problem solving skills, without being too overwhelmed.

Five levels of autonomy were identified for working with the micro:bit. This set might very well need to be changed or expanded with more levels. It can for instance be discussed if there are more layers outside of the one called assignment. As a completely autonomous assignment with the micro:bit still is an exercise limited to the hardware micro:bit, it is reasonable to say that there could be a level of hardware and maybe editor outside of the existing levels. This is however outside of the scope of this thesis.

Our experience is that it is beneficial to always provide some scope of autonomy in exercises, and never remove it entirely. As it was seen to pacify students when they did not have any way to affect the outcome of the exercises they were doing. In the scope of working with micro:bit, always providing some level of autonomy would translate into always allowing students to perform some level of customization in any exercise they are involved in. This relates to Papert(21) findings about students exposed to environments with creative freedom, have a higher tendency to learn the necessary knowledge in order to realize their ideas.

Different students are on different levels and require different scopes of autonomy in their exercises. As every student is different, with different skills and abilities to cope with making decisions, an exercise that works perfectly for one student might be overwhelming or boring to another student. This is in our opinion a challenge when working with a whole class. As it is hard to give every student individually adapted exercises with scopes of autonomy that matches their individual needs, teachers have to find exercises that can be given to the entire class. This means that a class with a wide span in individual progress, a single exercise can be perceived as anything from boring to useful to overwhelming. This can be tackled in various ways. One way is to try to minimise the skill span in the class, and unify the individual levels. This was partly what was attempted through the basic toolbox and analog workshops in this project. A second approach is to expand a single exercise's scope of autonomy

to be more flexible. In this way one single exercise can be given to an entire class, but different modifications or tips can be used to increase or decrease the scope of autonomy for the exercises, to better adapt it to individual students.

In Self Determination Theory(24) autonomy is mentioned both as a basic psychological need as well as the causality orientation described as "acting out of interest". This shows that the word autonomy can be a bit arbitrary, and hence might differ slightly from the way it is used throughout this thesis. The way it is used in this thesis is more relating to the amount of choices available to a student in a certain exercise situation. This positions our use of autonomy more as a term relating to the way a student's' context, the exercises, can be designed to satisfactorily support both the basic psychological needs of autonomy as well as competence. As it both relates to providing the student with enough choices to feel autonomous, yet not provide them with too many choices. As this runs the risk of having them feel that they lack the competence to succeed, which is how the basic need of competence is defined according to SDT.

What it means that a student is on a certain level or progress is not really understood at this point and we do not know how to measure it. So far the activity of adapting exercises scope of autonomy to fit the student's levels has been relying on the teacher's ability to customize exercises on the fly when helping students. Co-coding sessions were found helpful for trying to estimate how progressed students were in programming. This method is however limited by the fact that the most knowledgeable students are most probable to answer any questions the teacher might raise. This leads to the risk of having a few advanced students answering a teacher's co-coding questions, while others sit quietly without understanding. Yet it can be considered to be helpful that these students then at least get the opportunity to see how it is supposed to be done.

As for the deliverables we hope that the considerations about a basic toolbox, the scripted analog workshops, ideas about co-coding methods and other considerations will be able to inspire or be to some use for teachers who might be struggling with the new curriculum changes over the next few years.

7.3 Validity

Throughout the activities undertaken we realize that a majority of the participants had a positive bias towards curriculum changes and digitalization. Therefore there is no claim that the teachers represent an accurate image of the average mindset and motivation a teacher might have regarding programming. Even some students that were part of our research and data gathering led us to believe that they also were above average in being familiar with programming, although that was not the case entirely. Although it is easy to imagine that preparations for the digitalization has already started slowly and students being familiar with programming will soon become more common.

The type of research practiced has been solely qualitative. There was no way of testing our theories about our model, thus unable to gather large samples of data in order to generate statistics on the results. We believe that the nature of a qualitative research method allows researchers more room for subjectivity when deciding on the positives and negatives of a test. For the research to really be of significance there needs to be a thorough quantitative test were the progress can be measured and conclusions for large sample sizes can be made

7.4 Generalization

Throughout the project we have always related to the basic concepts of programming in our design process. These concepts permeate through all programming teaching activity and promote computational thinking and problem solving. Even though our model was made for micro:bit specifically there are many similarities that makes it versatile. Most of the platforms used to teach programming for primary school uses a block type editor, just like micro:bit. Other platforms can be better suited to teach some of the concepts in a vacuum, which is not the case for micro:bit, but the workflow still applies as it promotes a progress that aims to keep students at a balanced level of stimulation when learning. Since the underlying theories on the behaviour of students and how they learn is supported in the model, we believe the model can be claimed to have a wider applicability even though quantitative data to support the long term effects of using such model is non-existent in the thesis.

7.5 Future Work

During our early workshops we encountered the social phenomena of trends among the participating students. It began when a pair of students programmed a funny game and started playing, others took notice and wanted to join in. Soon the whole workshop had turned focus towards this game and everybody wanted to make it. During this period the students were highly involved, motivated and interacted with each other to a great degree, even helped one another. This behaviour has been seen multiple times throughout the project, and though it is positive, it seems hard to predict. This could be a topic to look closer at, we believe there is potential to capitalise on if the teacher can seize these moments when trends erupt and adapt the teaching to these opportunities. We decided to not investigate it further however, as the timeframe for this project did not allow us to dig into the field of social group psychology theories necessary for this kind of research.

The scope of autonomy model suggested in this thesis is merely an attempt to explain behaviours and phenomena that were observed throughout the project. A next step would be to see if it is possible to design a quantitative study that can validate or falsify the suggested correlation between the amount of choices presented in an exercise, and show the students potential to develop problem solving skills and feelings of being overwhelmed. As there was no time left to test the scope of autonomy model when it was finished, a next step towards validity would be to

collect quantitative data. The effect of such model need to be measured in a live setting where conclusions for large sample sizes can be made in order to truly gain any significance.

7.6 Ethical Issues

Possible ethical issues that may arise is the restricted use of images and video from workshops, as well as for presentation and reports due to the majority of test subjects is under aged. Images used may not contain faces of the participants or any other means that can identify them without their consent. Parental consent was required for kids participating in research activities such as recorded interviews or video observations. Another thing when working within a teacher's realm is not to discourage teachers by telling them "how to do their job" and merely take a low profile role providing guidance and suggestions and stay a humble observer.

8

Conclusion

Through an iterative design process a total of 21 workshop interventions were conducted to collect qualitative data and gain insights about users and their interactions with micro:bit teaching materials.

The aggregation of insights from throughout the project suggests that it is important for teachers to consider the amount of free choices that are given to students in any given exercise, when designing teaching materials for the BBC micro:bit, for training Swedish primary school students computational thinking skills. A model was created in an attempt to communicate these observed relationships between students learning potential, their risk of feeling overwhelmed and the amount of choices provided in exercises. More work will be needed to validate this model however.

A set of guidelines as well as a teaching approach was provided to further give more concrete answers to the research question: *What is important to consider when designing teaching materials with the BBC micro:bit for training Swedish primary school students computational thinking skills?*

Based on the model, the teaching approach and the guidelines, a few concrete implementation examples were created to meet the stated design goal to: *Support Swedish primary school teachers with teaching materials, based on the BBC micro:bit, that help them meet the programming requirements of the new curriculum changes.*

Bibliography

[1] Paulo Blikstein. Digital fabrication and 'making'in education: The democratization of invention. *FabLabs: Of machines, makers and inventors*, pages 1–21, 2013.

[2] Kate Williams. Literacy and computer literacy: Analyzing the nrc's "being fluent with information technology". *Journal of Literacy and Technology*, 3(1):1–20, 2003.

[3] Rachel Charlotte Smith, Ole Sejer Iversen, and Rune Veerasawmy. Impediments to digital fabrication in education: A study of teachers' role in digital fabrication. *International Journal of Digital Literacy and Digital Competence (IJDLDC)*, 7(1):33–49, 2016.

[4] Regeringskansliet. Stärkt digital kompetens i läroplaner och kursplaner - regeringen.se. `http://www.regeringen.se/pressmeddelanden/2017/03/starkt-digital-kompetens-i-laroplaner-och-kursplaner/`, May 2017. (Accessed on 05/23/2017).

[5] Skolverket. Förslag på en nationell it-strategi för skola och förskola. `http://www.skolverket.se/om-skolverket/publikationer/visa-enskild-publikation?_xurl_=http%3A%2F%2Fwww5.skolverket.se%2Fwtpub%2Fws%2Fskolbok%2Fwpubext%2Ftrycksak%2FRecord%3Fk%3D3621/`, 2016. [Online; accessed 2-December-2016].

[6] Steve Furber et al. Shut down or restart? the way forward for computing in uk schools. *The Royal Society, London*, 2012.

[7] Neil CC Brown, Sue Sentance, Tom Crick, and Simon Humphreys. Restart: The resurgence of computer science in uk schools. *ACM Transactions on Computing Education (TOCE)*, 14(2):9, 2014.

[8] on the About the BBC Blog Head of BBC Learning, Sinead Rocks. Bbc micro:bit, groundbreaking initiative to inspire digital creativity and develop a new generation of tech pioneers. `http://www.bbc.co.uk/mediacentre/mediapacks/microbit/`, 2016. [Online; accessed 2-December-2016].

[9] Scratch - imagine, program, share. `https://scratch.mit.edu/about`. (Accessed on 05/28/2017).

[10] Arduino - introduction. `https://www.arduino.cc/en/Guide/Introduction`. (Accessed on 05/28/2017).

[11] Makey makey. http://makeymakey.com/faq/. (Accessed on 05/28/2017).

[12] Raspberry pi faqs - frequently asked questions. https://www.raspberrypi.org/help/faqs/#introWhatIs. (Accessed on 05/28/2017).

[13] Picademy - free professional development from raspberry pi. https://www.raspberrypi.org/picademy/. (Accessed on 05/28/2017).

[14] Guide – quirkbot basics — quirkbot. https://www.quirkbot.com/guide-quirkbot-basics. (Accessed on 05/28/2017).

[15] Join the largest learning event in history, 5-11 december 2016. https://hourofcode.com/se/en. (Accessed on 05/28/2017).

[16] Computing at school. https://www.computingatschool.org.uk/about. (Accessed on 05/28/2017).

[17] Fab foundation – what is a fab lab? http://www.fabfoundation.org/index.php/what-is-a-fab-lab/index.html. (Accessed on 05/28/2017).

[18] Techshop is the world's first open-access workshop – what do you want to make at techshop? http://www.techshop.ws/. (Accessed on 05/28/2017).

[19] About our project – makerskola. http://makerskola.se/about-our-project/. (Accessed on 05/28/2017).

[20] Digitalverkstan — vad är digitalverkstan? http://digitalverkstan.com/about. (Accessed on 05/28/2017).

[21] Seymour Papert and Idit Harel. Situating constructionism. *Constructionism*, 36:1–11, 1991.

[22] John AC Hattie and Gregory M Donoghue. Learning strategies: a synthesis and conceptual model. *npj Science of Learning*, 1:16013, 2016.

[23] Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[24] Edward L Deci and Richard M Ryan. Self-determination theory: A macrotheory of human motivation, development, and health. *Canadian psychology/Psychologie canadienne*, 49(3):182, 2008.

[25] Marylène Gagné and Edward L Deci. Self-determination theory and work motivation. *Journal of Organizational behavior*, 26(4):331–362, 2005.

[26] Frédéric Guay, Catherine F Ratelle, and Julien Chanal. Optimal learning in optimal contexts: The role of self-determination in education. *Canadian Psychology/Psychologie canadienne*, 49(3):233, 2008.

[27] Jeannette M Wing. Computational thinking. In *VL/HCC*, page 3, 2011.

[28] Shuchi Grover and Roy Pea. Computational thinking in k–12 a review of the state of the field. *Educational Researcher*, 42(1):38–43, 2013.

[29] Harvard Edu. Computational thinking with scratch. http://scratched.gse.harvard.edu/ct/defining.html/, 2016. [Online; accessed 2-December-2016].

[30] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, pages 1–25, 2012.

[31] CAS Barefoot. Computational thinking. http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking//, 2014. [Online; accessed 2-December-2016].

[32] Stanford. Empathy map. https://dschool.stanford.edu/groups/k12/wiki/3d994/Empathy_Map.html/, 2016. [Online; accessed 29-Novemberr-2016].

[33] T Zorn. Designing and conducting semi-structured interviews for research. *Waikato Management School*, 2008.

[34] Entrance & exit tickets | the sheridan center for teaching and learning. https://www.brown.edu/about/administration/sheridan-center/teaching-learning/effective-classroom-practices/entrance-exit-tickets. (Accessed on 05/28/2017).

[35] A. Cooper, R. Reimann, and D. Cronin. *About Face 3: The Essentials of Interaction Design*. Wiley, 2007.

[36] LUMA Institute. *Innovating for People: Handbook of Human-Centered Design Methods*. LUMA Institute, 2012.

[37] Maker movies. http://makermovies.se/movies. (Accessed on 05/12/2017).

[38] Kom igång med programmering – kodboken. https://www.kodboken.se/start/skapa-spel/lekar-och-ovningar/robotkompis. (Accessed on 05/05/2017).

[39] Botrace. http://gyulai.se/botrace/. (Accessed on 05/05/2017).

[40] Programmera mera - ur skola. https://urskola.se/Produkter/196673-Programmera-mera/Visa-alla. (Accessed on 05/05/2017).

[41] Gustav Fridolin and Mikael Damberg. Vårt löfte till barnen – mer teknik i skolan. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/, 2016. [Online; accessed 2-December-2016].