Metaphor-based tangible toolkit for programming education

Master's Thesis in Computer Science

Mustafa Numic

June 4, 2020
Halden, Norway

Østfold University College www.hiof.no

Abstract

Metaphors have been used previously in education with positive learning effects, as well as tangible programming. But when combining these two methods into one method or tool it could bring positive learning effects on beginner programmers at a college course. The research questions for this master thesis is "How can I make a tangible metaphor to describe a programming concept?" and that brings the sub question "What impact does the tangible metaphors have on student understanding?". Based on the user testing and interviews conducted, it shows that the metaphor based tangible programming toolkit have a positive impact on students understanding of programming concepts

Keywords: Tangible toolkit, Metaphor based, Education, college

Acknowledgments

I would like to thank my supervisor Tore Marius Akerbæk for for providing feedback on the project and the write up of the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

Metaphors are a way of introducing something new by describing it with something that the person already knows, therefor there must be a relationship between the new idea and the metaphors[Sajaniemi et al., 2007].

There have been numerous studies and articles on using metaphors in teaching both programming courses and other school courses, but we could not find any articles or studies that have used tangible toolkits using metaphors in teaching at this time. Most studies that are published have used metaphors for teaching programming to children ages 9 to 12, only a couple of people have tried using metaphors in higher education at college level, and only on textual metaphors.

Based on the study done by [Hidalgo-Céspedes et al., 2018] using metaphors in educational setting do bring positive outcome in student participation and understanding.

I want to test if tangible metaphors for teaching programming can be made and have positive effects on students understanding of programming concepts. My goal is to use daily used metaphors and allegories to teach students programming concepts in courses at the university level.

Based on the study done by [Salleh et al., 2013] concluded that programming tools that uses visualization approach were much more preferred by students, meanwhile game related elements makes the learning of programming much more interesting. They also found that the complexity in programming is the main reason for lack of motivation among students. When they reviewed how much research has been done on using different tools for teaching programming they found that 12 of 45 research papers used visualization and only 2 of 45 used physical tools. I want to try and combine that, with using metaphors as visualization and that it is also a tangible physical tool for teaching programming [Salleh et al., 2013].

In the study done by [Robins et al., 2003] called Learning and teaching programming: A review and discussion, he states that introductory programming courses are hard and has one of the biggest dropout rates. In their article they also go through some teaching and learning methods, where they conclude that alternatives to the conventional curricula shows promise but that none of them have yet come to dominate the theory and practice of programming pedagogy. My approach is to supplement the conventional curriculum, by not replacing the methods used today to learn programming but by using the same methods and just adding the toolkit as a way to describe and learn different programming

concepts.

The project is in cooperation with the Østfold university college in Halden Norway.

1.2 Research question

1.2.1 Research question/Problem statement/Objectives

As mentioned earlier we could not find any studies or articles that have tried implementing tangible metaphors in higher education for teaching programming concepts to students. Since most studies that have used metaphors in programming education have tried to implement metaphors in students aged 9 to 12, we chose to focus on students at college level, since older students have a different understanding of metaphors than kids. A 20 year old student will have different concepts of metaphors than a 9 year old child. For example if using a car as a metaphor a person who drives will have a different understanding of the metaphors than a child that does not drive.

In our study we want to make a toolkit with tangible metaphors that can then be used to teach programming concepts to students. Based on the studies done by others on using metaphors in programming education[Mckay, 1999],[Lynch and Fisher-Ari, 2018], we believe that using tangible metaphors where students gets to work with their hands and move objects to conceptualize programming concepts, will benefit students understanding of basic programming. The first research question for this project is:

RQ 1 *How can I make a tangible metaphor to describe a programming concept.*

It is also important to measure the impact the tangible metaphor has on student understanding of programming concept. For measuring the students understanding of programming concepts after using the tangible metaphor toolkit we need to have a second sub question for the first research question which is:

RQ 1.2 *What impact does the tangible metaphor have on student understanding?*

1.3 Report Outline

Chapter one describes the background and motivation behind the master thesis. Chapter one also describes the research questions. And lastly it will go through the report outline.

In chapter two I analyze what methods will be used to answer the proposed research questions in chapter one. First I go through the literature review and how I choose which articles are relevant for my master thesis. After going through the literature review, chapter two describes the research method used for prototyping and user testing.

In chapter three I will go into more detail on the chosen research topic and information on related work articles I have found while researching the topic. Where the first section will describe the research topic, and the second section which will describe the related work and what I got from reading the articles. The related work section will be split into 4 subsections which will be going to go into more debt on using metaphors in teaching settings, what programming concepts previous articles have studied, metaphors that have been used for different programming concepts and lastly using tangible programming in educational settings.

Chapter four describes the design, prototyping and testing process used. The first part of chapter four goes through choosing a programming language and a programming concept. After that the chapter describes in depth the programming concept that I choose to focus on in this master thesis. Chapter four also describes the whole process of making the prototype and also the user testing, when the prototype was finished.

In chapter five I analyze and discuss the process and work done. Also in chapter five I analyze if I have answered the proposed research question.

The last chapter, chapter six I analyze the future work that can be done with this concept.

Chapter 2

Method

2.1 Methodology

To answer the proposed research questions I plan on using previous work to find some usable metaphors. Some of the textual metaphors used in previous studies can be directly translated to tangible metaphors others, need some tweaking to work with being tangible, while also being able to be used in a classroom setting. The methodological approach I plan on using in this project is a Qualitative approach. Since I could not find any previous work that has tried doing metaphor based tangible toolkits, this project will be exploratory in making the actual prototype. But there have been studies that has used textual metaphors for programming which I can base my metaphors on, these metaphors will be discussed in the chapter Related work.

For designing and producing the working prototype I plan on using the research through design method, which means that I design and make the prototype without involving the user. When I have a prototype ready then I plan on testing it with users. I want to use field testing with semi structured interviews after the study. The field test is planned to be in an introduction to programming course in Østfold university college.

2.2 Literature review

I will perform a literature review early in the process of the thesis, to see what has been done before so I can build my thesis on a foundation of knowledge on the topic. The purpose of the literature review is so I justify my research and how I can add more knowledge on the topic. Lastly the literature review is meant to place my research within the context of the existing literature done previously by other researchers. For the literature review I used the databases Oria and ACM, with the search frases shown in the table below. All search frases had the filters: Peer reviewed, English language and Journal article. I also used some snowballing method for finding some of the articles.

Search results			
Search no.	Search words	Oria	ACM
s1	Programming metaphors	5,226	17,344
s2	education metaphors	55,889	6,875
s3	Tangible toolkit for programming education	456	18,697
s4	s1 and s2	3,662	18,466

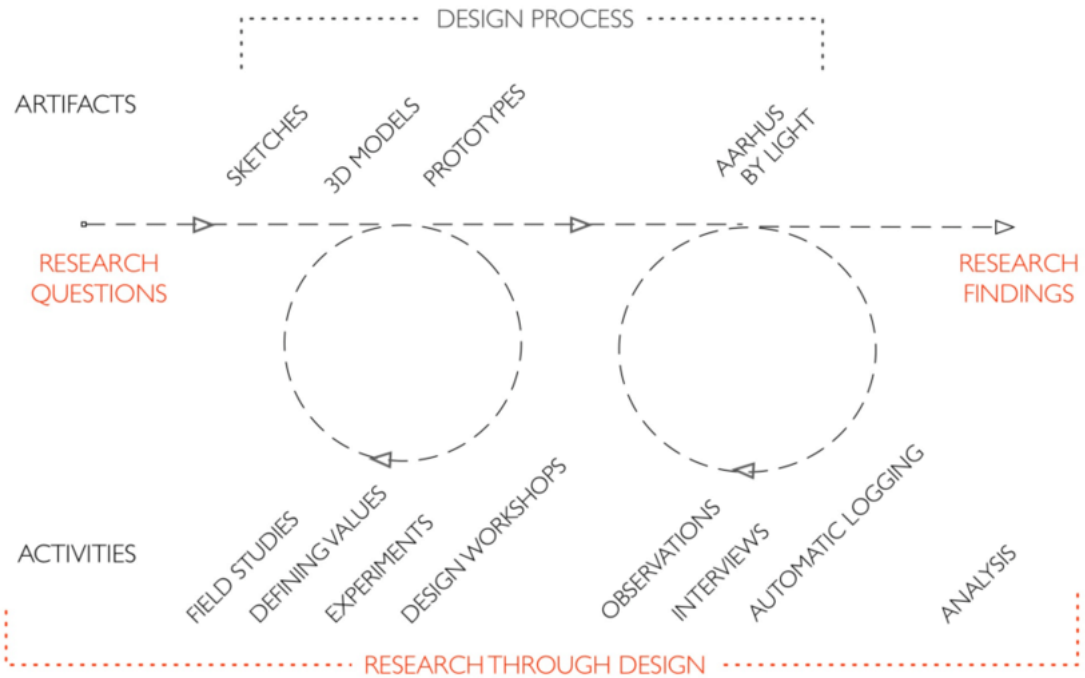
Most of the search strings produced a lot of result, so I had to use some filters to filter out the less relevant articles for my thesis. For limiting down the results gotten from the database the first filter used was that it had to be peer reviewed articles and that it had to be a journal article and it had to be written in English. After that when searching for programming metaphors, I used the filter for filtering on the subject which was in this case programming. that cut it down from 5.226 result down to 51. Lastly I go through and read the abstracts and based on that I decide if its worth to read the whole article, if I think that it can add some useful information to my master thesis.

When searching for education metaphors I used the same filters in the start as the first one, Peer reviewed, journal article and English language. After adding all those filters it still had quite a few result, so adding a filter on the subject helped eliminate a lot of the result that did not produce any useful information. I filtered on the subject education and then also on the subject metaphors in education, that produced 125 results which was a lot easier than going through 55 thousand. For choosing which was relevant I read the abstracts and decided upon reading the abstract if it was relevant for my thesis.

2.3 Research through design

Research through design is primary used in academic work in the design community. In the encyclopedia of human computer interaction chapter 43, they first describe what is research and what is design. Their working definition of research is doing work with the intention to produce knowledge that will be used by other, while the definition of design is doing work with the intention to produce a solution to improve a situation. In chapter 43 it states that research through design combines research and design by making a prototype that plays a central role in generating knowledge[STAPPERS and GIACCARDI,].

Figure 2.1: Research through design



In research through design the designers makes the prototype based on the knowledge that has been produced in previous works done by other researchers and by doing other methods for designing the prototype like sketches or storyboards. Then they use the prototype they have made to do experiments and make appropriate changes to the prototype before using it to generate more knowledge on the proposed research question, by doing observations and interviews, as seen on figure 2.1.

In the study done by [Zimmerman et al., 2007], exploring using research through design as a method for interaction design project. He found five main benefits from using research through design as a method for interaction design project. First he found that research through design lets the design community explore problems that can not be easily addressed with science and engineering methods. The second benefit they found was that it feeds technology opportunities and unexpected behavioral to the engineers and behavioral scientists, which then leads to more research done. The third benefit they found by using research through design was that it provides a new method to transfer knowledge that was acquired in HCI research to the HCI practice community, which then hopefully will be transferred to products in the real world. Second to last benefit they found was that it allows designers to make research contributions by reframing problems and making the right thing. The last benefit they found was that it motivates the HCI community to discuss the impact the research might have on the world.

2.4 Storyboarding

Storyboarding is sequential art that visualizes a story. This method came from motion picture productions. Storyboarding is a good way of illustrating a idea or concept. By

making a story board of the idea, the designers can see if the concept will work. Storyboarding can also help the designers identify errors at a early stage in the designing process, it also helps the designers to walk in the users shoes [Truong et al., 2006]. Storyboards are used in a lot of different activities, like when designing a new technology, a storyboard usually illustrates the envisioned scenario of how a application feature works. And in interaction design scenarios, the storyboards shows how the users would interact with the proposed system.

2.5 Prototyping

Prototyping is a important part of the designing process, because it allows the designer to test and make changes to the ideas. A prototype is a model of the proposed solution which can be used to test or validate different aspects of a idea. Prototyping is used to test the idea before to many resources gets used, by providing different tools and approaches to test a idea. Prototypes can be as simple as something made from paper to illustrate a idea or a more formed and detailed prototype to be used in pilot studies. A prototype does not need to be the full product. Prototyping is about bringing conceptual ideas to be tested in the real world. Research done early in the design process can bring biases towards ones ideas, by making a prototype one can test it to reveal biases towards your idea [Rikke Friis Dam,]. Prototypes can be split into 3 forms of prototyping, Low fidelity, medium fidelity and high fidelity. Low fidelity is used early in the design process and focus more on the way of using the proposed product. Low fidelity are often paper prototypes which lets the designer make changes cheap and efficiently in the beginning. Medium fidelity prototypes is where the product has more practical functionalities, but is stil limited. Medium fidelity prototypes are often built upon user scenarios and story boarding. High fidelity prototypes are a lot closer to the final product. High fidelity prototypes gives a realistic experience of the product with actual functionalities.

The intended method I will use for making the prototype is research through design. In this method I will make a prototype based on previous work which I will discuss in chapter Related work, and by using story boarding for conceptualizing the metaphors I plan on making tangible. The storyboards will help me visualize the metaphors I will be making tangible, by illustrating what parts the prototype need to have to be able to be used later in the testing phase. Then when I have a working prototype I will use it for the research by doing field studies for making changes to the product as seen on the figure 2.1. The prototype will also answer the proposed research questions in my thesis.

2.6 Data collection

There are several different method for collecting data when testing the prototype. Here I will be discussing the different methods that could be used.

2.6.1 User testing - observation

User testing is a important part of the design process. One important reason for user testing is so that you can make something that is relevant and the users wants to use. User testing is used to gain knowledge about the users you are designing for. People expect product to be easy to use and learn, user testing can help designers achieve this. If the

designers wants users to use their products, their user experience has to be good or the users will move on to another product[Mortensen,]. There are 3 kinds of observations that can be done when testing the prototype [McLeod, 2015]

The first is controlled observations. A controlled observation is likely to be carried out in a place where the researchers have chosen it to be, and at what time with which participants. In controlled observations all the variables are controlled by the researchers and the participants knows that they are being watched.

The second method is naturalistic observations, which means that the researchers are observing the spontaneous behavior of the participants in the natural surroundings. In naturalistic observations the researchers does not have any control of the variables, and only record what the participants does.

The last form of observations is participant observations. Participant observations is a combination of both previous forms. In participant observations the researchers joins the group they are observing to get a deeper understanding of the participants. Participation observations can either be where the participants thinks that the researcher is a genuine member of the group or where the researcher revels who they are and asks for permission to observe.

2.6.2 Interviews

A interview is a good way to have a conversation with a participant but with a purpose. Interviews gets more personal ideas than a survey, by going into more detail and understanding what the users wants. interviews can be useful for collecting data regarding the users satisfaction and thoughts after testing a product. There are four different kinds of interviews for collecting data when trying the prototype [McLeod, 2014].

The first is structured interviews. In a structured interview the researchers have a set of prepared closed ended questions that the participants has to answer, and they do not deviate from the planned questions.

The second form of interviews is a unstructured interview. In a unstructured interview the researchers has not prepared any questions in advance, the researcher tries to have a natural conversation with the participants and tailors the questions to the participants specific experience.

Then there is semi structured interviews which is a combination of the two methods previously mentioned. In a semi structured interview the researchers does have some questions prepared in advance, but can also deviate from the questions to explore each participant and their experience.

Lastly there are focus group interviews. In a focus group interview, a group of participants are interviewed together with a interview moderator which is there to make sure the group interacts with each other and that they do not drift of topic.

2.7 User testing

For my project I will be using observations for gathering information on what the user think about the experience when using the prototype combined with using semi structured interviews to answer my research questions. By doing semi structured interviews I can have some main questions planned to guide the conversation and keep the participants on topic which would not be able to do with a structured interview. Since I had planned a

few questions in advance and had some points I had to go though with the participants I could not conduct a unstructured interview.

The user testing will be conducted on the target group which is beginner programmer students at a introductory programming course. After each student tries the prototype, I will ask them if they want to answer some questions, that are discussed in the sub chapter Questions. Ì will have some main questions prepared beforehand, and by doing semi structured interviews I can go more in depth on the answers that the users give. The semi structured interviews gives me the opportunity to ask the users more questions that arise while conducting the interviews. The results acquired from the user testing and semi structured interviews will then be discussed in chapter 4.5.3 results.

Chapter 3

Related work

3.1 Related work

3.1.1 Using Metaphors in teaching

The idea of using metaphors in teaching has existed for a long period of time[Waguespack, 1989]. Article Visual metaphors for teaching programming concepts is over 30 years old. In this article Leslie Waguespack used metaphors for visualizing programming concepts in a introductory programming course using pascal. The programming concepts he chose to use was data types, variables, arrays, records, files, modules, module interfaces and parameter passing. Waguespack, Leslie J. could not confirm if using metaphors for teaching programming concepts did improve student understanding, because he could not isolate the metaphors from the other course components.

Heather Lynn Lynch and Teresa Renae Fisher-Ari did a study in 2018 about using metaphors in college teaching, and the effects of it. While the study does not directly relate to programming and teaching of programming concepts, they did find out some interesting things about using metaphors for teaching. This study was done on students studying teaching. In their study they made the students write metaphors about the subject curriculum. What they found out was that the metaphors the students wrote helped them track when the class or specific individuals needed further support. And that by using metaphors they got more student participation[Lynch and Fisher-Ari, 2018].

The study done in 2018 on the effects of oral metaphors and allegories on programming problem solving, concluded that they did not find any difference between these two types of teaching, but that further testing did need to occur for them to be definitive. They also stated that the lack of significant difference between allegories and oral metaphors are the most common recurrent result reported by previous studies. Therefor we want to try using tangible metaphors where the student actually does something that is supposed to be a metaphor that represents the programming concepts[Hidalgo-Céspedes et al., 2018].

Based on a study done in 2007 by Jorma Sajaniemi et al they concluded that there are positive effects of using metaphors for learning complex tasks, but using metaphors in simple tasks shows no effect and since programming is a complex task, using metaphors can produce positive effects[Sajaniemi et al., 2007].

Back in 1999 Gregory Jay Gassner did a study on using metaphors in high performance teaching. He concluded that using metaphors increases students rate of learning and is a good way for students to acquire new athletic skills. In the study Gregory Jay Gassner says

that metaphors has a enhancing property when used in teaching, because the student has to interpret the metaphors and act accordingly[Gassner, 1999]. While Gregory Jay Gasnner did the study on using metaphors for high performanfce teaching and coaching, Elspeth Mckay did a study on Exploring the Effect of Graphical Metaphors on the Performance of Learning Computer Programming Concepts in Adult Learners. divided a focus group into two parts, one which only got textual learning and the other part that got textual and graphical learning. He found that the students that had both methods learned the most, and that the answers where more innovative rather then the answers from the students that got only the textual treatment. The answers from the students that only got the textual treatment often resembled the text from the book[Mckay, 1999].

3.1.2 Programming concepts

First we have to identify what programming concepts there are, and which we need metaphors for. The programming concepts used in A methodology proposal based on metaphors to teach programming to children [Pérez-Marín et al., 2018], which tries to find a method to teach young children programming. Their approch in this study was using metaphors, and they suggested to start with these programming concepts.

- Program
- sequence
- memory
- Variable
- Input and output
- Conditional
- Loop

This article was written with focus on children that are completely new to programming. The study a methodology proposal to teach programming to children also used the same programming concepts, since our target group are older students but that are also completely new to programming these are some good concepts to start with[Pérez-Marín et al., 2018].

In the study done by Jeisson Hidalgo-Céspedes et al done in 2018 they used 3 programming concepts that they then divided into either one metaphor or multiple metaphors. The programming concepts used are:

- computer programming
- event-driven programming
- application user interface

These programming concepts are described with multiple metaphors. In this project we plan on teaching the basic programming concepts which can be described with one single metaphor, while in this project they used more than one metaphor[Hidalgo-Céspedes et al., 2018].

Jorma Sajaniemi et al did a study on using animation metaphors for object-oriented concepts. In the study they animated programs with metaphors, So that its easier to

read. They found out that the animation did not scale to larger programs, so they wanted to use it in elementary programming courses. The Concepts they chose to use are [Sajaniemi et al., 2007]:

- Classes
- Object
- Method invocation
- Parameter passing
- Return value
- Object reference
- Garbage collection

These kind of programming concepts are what we want to focus on with our project.

A study done by Irina Rogozhkina et al in a Moscow kindergarten, where they wanted to teach kindergarteners programming concepts by making them program a robot using pictograms. They wanted to teach pre-schoolers some fundamental programming concepts like:

- Linear programs
- Subprograms
- Programs

[Rogozhkina and Kushnirenko, 2011].

3.1.3 Metaphors used for programming concepts

Before being able to proceed with making a prototype we need to find out which metaphors we can use for different programming concepts. In the study done by Diana Pérez-Marín et al they used the metaphor recipe for describing a program or a sequence. The metaphor they used to describe input and output, are pc screen for output and keyboard for input. Lastly they used a child making decision for conditional and computer repeating instructions x times for loop. These metaphors are designed for children, but since this project is targeted more for a older audience, some of the programming concepts are going to be explored and altered to better suit the target group.

A study from 2007 on using animation metaphors for object oriented concepts,[Sajaniemi et al., 2007]. They focused on programming concepts that are more complex than input and output that are meant for children. They chose to use blueprint as a concept for classes. This metaphor describes really good what a class is and can be directly translated to something tangible. For the concept of object they chose to use watch panel. They described method invocation with workshop, they used envelope for describing parameter passing. For return value they chose to use the metaphor workbench, and pennant for describing the concept of object reference. Lastly they chose to use a garbage vehicle for describing the concept of garbage collection.

3.1.4 Using tangible programming for education purposes

Danli Wang et al did a study in 2013 on using tangible blocks for teaching children aged 5 to 9 about programming. They made a computer program which then needed to be solved by the student by combining different blocks to finish the task presented on screen. They concluded the study with finding that students focused much more on programming then different things around them. They compared their method with the previous method used to teach children programming and found that their new method had better effects and that children learned more when having fun [Wang et al., 2013].

The project done by Kunal Chawla et al found out that their tangible programming toy attracted more than only children to try it out. They made a toy for teaching young children programming with combining blocks. They did not do any field study or user testing, but they managed to showcase their prototype at a expo and found that different people came by to test it out. All from children to adults and older people who had no experience programming found the toy very interesting [Chawla et al., 2013].

A study done by Michal Gordon et al found out that children from 4 to 8 years old, engaged highly with the social programming robot they made. Children got a better understanding of how programming a robot is and learned some basic programming concepts [Gordon et al., 2015].

Based on study done in 2019 by Theodosios Sapounidis et al on tangible and graphical programming with experienced children. They tested the difference between graphical programming and tangible programming on children aged 8 to 9 and 12 to 13. They found that the children preferred the tangible toolkit, and that they had more retention of the knowledge. They also found that the children had less problems coming back to the tangible toolkit and just pick it up. Something that is also interesting is that children did colabarate more when using the tangible programming toolkit [Sapounidis et al., 2019].

Chapter 4

Design / Prototyping

4.1 Choosing a programming language

The long term intention of the tangible programming toolkit is to offer multiple programming languages as the output, with the respective language syntax and structure being displayed on-screen as the user administers the tools in the toolkit. Given the time frame of a master thesis, I will focus on one programming language as the output from the prototype. [King, 1997] suggest Java as an introductory programming language, because of Low cost, the tools needed to build and test java programs are available without charge. He also suggests Java because of student enthusiasm, he states that java has gotten so much publicity that students are going to be much more exited to learn Java. He also states that Java is also suitable for advanced courses later on when the student has learn the basics. He recognizes the importance of introducing students to the object-oriented paradigm early in their learning of programming. He comes to the conclusion that Java has a significant advantage to being used as a introductory programming language, because the students get introduced to the object-oriented paradigm without being exposed to the complexity of C++. [Kruglyk and Lvov, 2012] did a study on what language is best to replace pascal as a introductory programming language. In his study he compared Pascal with C, Java, Python, C++, Object Pascal, PHP and Javascript. He concludes that the only draw back for using pascal in introductory programming courses is its use in real world projects. He states that C and Java are similar languages and that they booth are very important in the IT field but also hard for beginner programmers to learn, he also concludes that Javascript is not suitable to teach beginner programmers for its peculiarities of use. In his study he comes to the final decision that python is a suitable replacement for pascal as a introductory programming language, because of it is a high level programming language for joint purposes, with an emphasis on developers productivity and readability of code. [Vujošević-Janičić and Tošić, 2008] states in their article that the choice of the first programming language used in introductory courses is still challenging. They also state that students should first learn object oriented programming since it is a lot harder to shift paradigm, therefor it would be smart to start with a object oriented programming language to avoid this shift. They found that the idea of choosing a programming language to be used in introductory courses is going to always be discussed and that there have been many case studies that concludes that support different languages. They state that the most important part of a introductory course is to focus on the general programming ideas and programming concepts. That is why I want to make

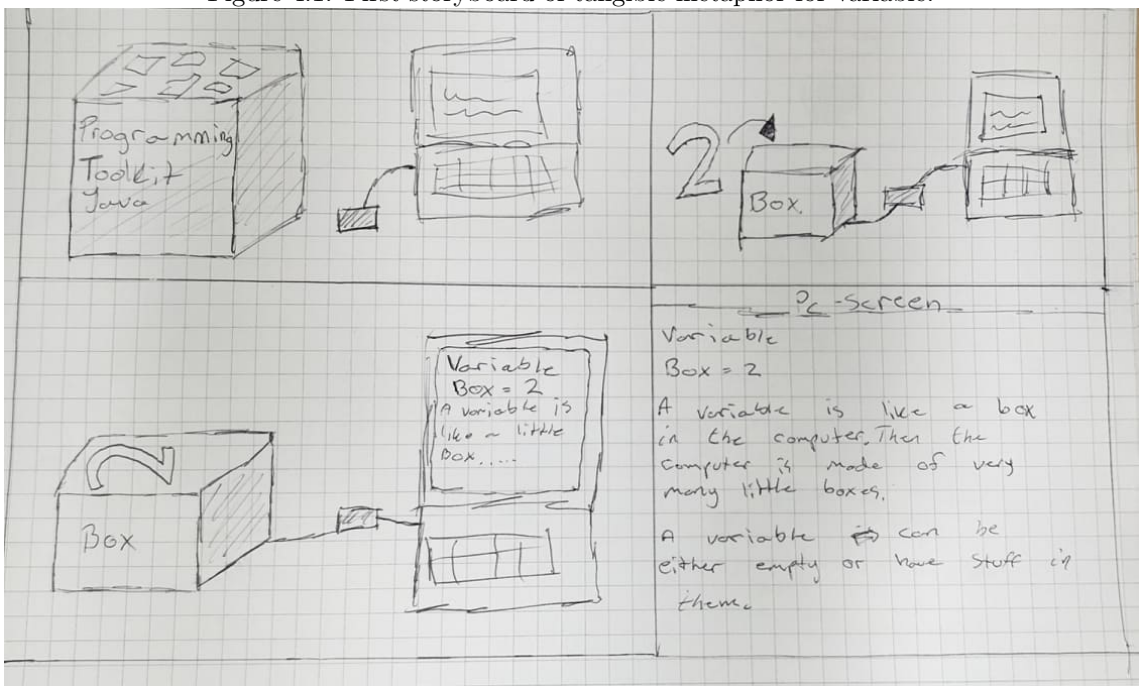
a toolkit that can offer multiple programming languages as the output, but based on the time frame of a master thesis I will focus on the object oriented language Java.

4.2 Choosing a programming concept.

A lot of the studies I have found proposed using variables as a programming concept to start with, since it is one of the first concepts a student learns when starting a introductory course. A common metaphor for variables in these studies are a box. [Waguespack, 1989], [Pérez-Marín et al., 2018]. Booth Waguespack and Perez-Marin et. al. used a variable as one of their programming concepts. The variable as a programming concept is very important and it is one of the first thing a student learn in a introductory course. Variables can be described as the backbone of any programming language. [1] Based on that the variable is one of the first programming concepts a student learn and that it has a metaphor that is well tested in previous studies, I choose to use the variable as the programming concept I will focus on in my master thesis.

4.2.1 Variable

Figure 4.1: First storyboard of tangible metaphor for variable.



In this concept the variable is the box. A box is used as a metaphor in a couple of the studies done on using metaphors for teaching programming concepts.[Mckay, 1999] As shown in the storyboard the student can choose from a toolkit with different metaphors for describing programming concepts. The student then chooses the metaphor for a variable. When the student has chosen the variable metaphor, he can then proceed to plug it in to

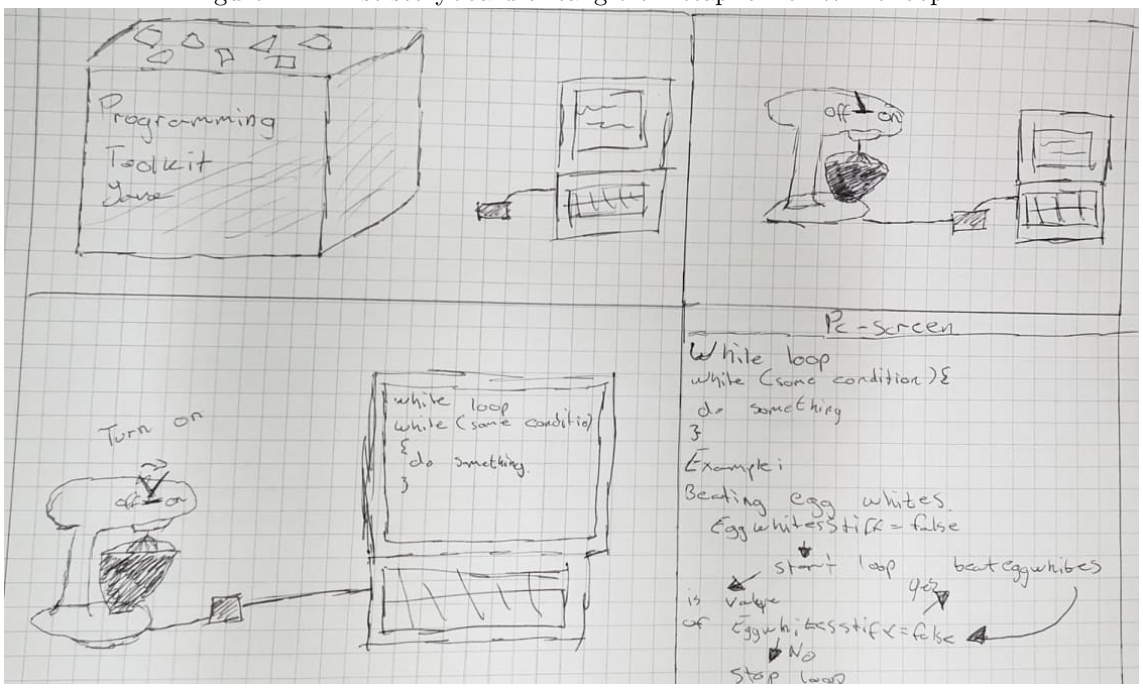
[1] https://howtoprogramwithjava.com/programming-101-the-5-basic-concepts-of-any-programming-language/

the pc. When the box has been plugged into the pc the student can name the variable by typing the name of the variable on the keyboard. Then the student can put a number in the box from 0 to 9 which is made as a token. When the student has put the number in the box, the corresponding code and a description.

4.2.2 While-loop

The next concept I tried making a metaphor for was a while loop. Here the student can pick up a little version off a kitchen mixer, when the student plugs it into the computer he gets to choose for how many iteration the loop Will run. then the loop starts a little motor in the kitchen mixer that runs for the specified amount of turns. This action can also be related to when people are beating egg whites in the kitchen. The kitchen mixer continues to run until the egg whites are done, then the person turn the mixer off. When the mixer starts to run through the loop the corresponding code will come up on the pc screen, as seen on the storyboard below.

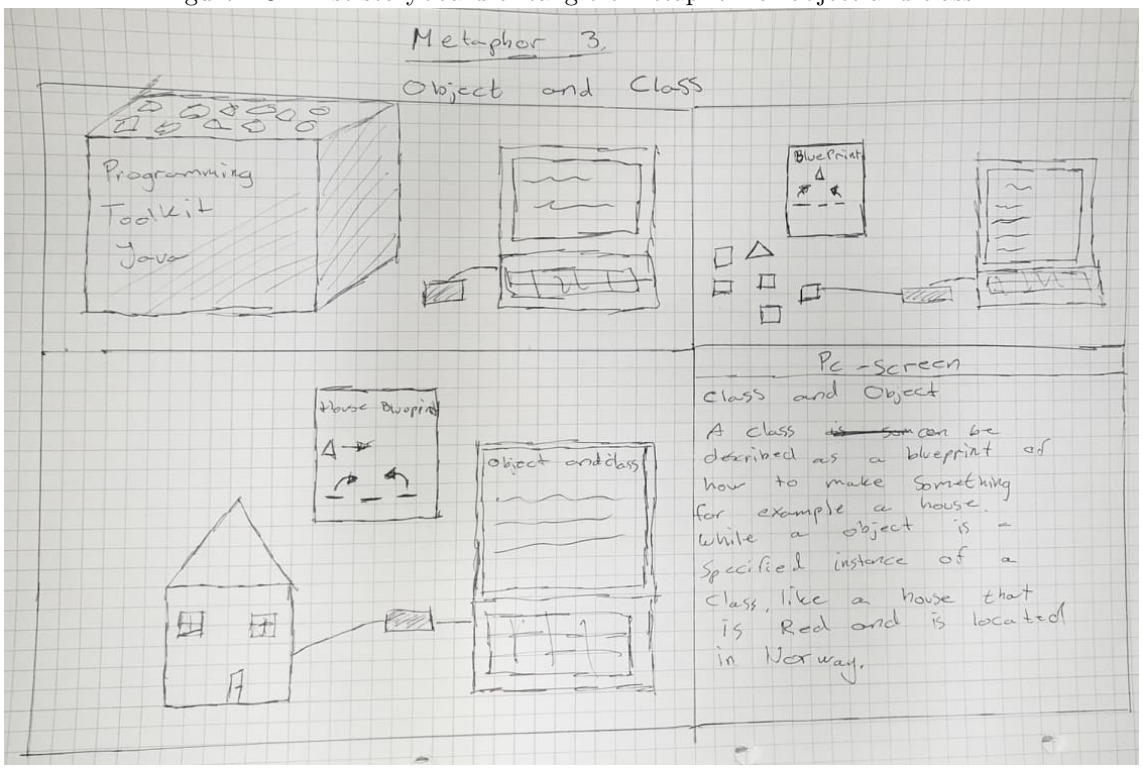
Figure 4.2: First storyboard of tangible metaphor for While loop.



4.2.3 Object and Class

The last concept I made a metaphor for was for object and class, as seen in the storyboard below. In this concept the student gets a blueprint, that is meant to describe a class. The blueprint will show the student how to build a house. The house is meant to describe a object. In the toolkit for this metaphor there will be different kinds of walls and door, so the student can choose the color of the house. When the student is finished with the house the corresponding code will come up on the computer. Each block of the house will connect only to the corresponding blocks, so when all the blocks are put in the right place it will send a message to the pc that there is a instance of that class.

Figure 4.3: First storyboard of tangible metaphor for object and class.



4.2.4 Choosing one concept

After I made all three of the metaphors describing programming concepts I choose to focus on one concept and rather go more in depth in to one concept and metaphor. I choose to go in depth on the variable metaphor and concept because the variable is where most students starts their learning, also the use of a box or envelope as a metaphor for variable has been well documented as a good metaphor in studies done previously by other people.

4.3 Java variable in depth

I choose to focus on the variable concept and metaphor, that means that I had to go more in depth on what a variable is and what it is made up off.

4.3.1 Variable types

After some research on variables in Java I found that there are two types of variable, there are primitive and non-primitive variables[2]. I used the Java variable documentation to find what types of variable and operators there are in Java. Primitive variables are the most basic data types in java. The primitve data types are:

Primitive variables					
Category	Types	bits	Minimum value	Maximum value	Example code
Integer	Byte	8	-128	127	Byte b = 50;
	Char	16	0	$2^{16} - 1$	Char c = 'A';
	Short	16	-2^{15}	$2^{15} - 1$	Short s = 50;
	Int	32	-2^{31}	$2^{31} - 1$	Int I = 50;
	Long	64	-2^{63}	$2^{63} - 1$	Long l =50L;
Floating-point	Float	32	2^{-149}	$(2 - 2^{-23}) * 2^{127}$	Float f = 50f;
	Double	64	2^{-1074}	$(2 - 2^{-52}) * 2^{1023}$	Double d = 50.50;
Other	Boolean	–	False	True	Boolean b = true;
	void	–	–	–	–

The other type of variable are non primitive variables. In java non primitive variables are array, string, class and interface. A string data type is used to store consecutive characters, or whole sentences. A example of this is:
String Example = "This is a string example";
A non primitive variable is also known as a reference data type, since java does not hold onto the value of the variable but the reference to the value. The other non primitive data types are a array, class and interface. A array is used to store multiple data types in consecutive manner. The size of the array can be specified by the programmer. The main difference between primitive and non primitive data types in java is that primitive data types are predefined while non primitive data types are declared by the programmer except for string. A non primitive data type can be used to call methods to perform certain operations, while primitive data types can not do that.

[2]https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op2.html

4.3.2 Java operators

Java has 6 categories of different types of operators that can be used together with variables. The first category of operators are arithmetic operators. Arithmetic operators are used in mathematical expressions. In the table below A = 7 and B = 3, The operators are:

Arithmetic operators		
Operator	Description	Example
+(Addition)	Adds together two values on either side of the operator	A + B = 10
-(Subtraction)	subtracts the right value from the left value	A - B = 4
*(Multiplication)	Multiplies the values on either side of the operator	A * B = 21
/(Division)	Divides the left value from the right value	A / B = 2.3
%(Modulus)	Divides the left value from the right value then returns the remainder	A % B = 1
++(Increment)	Increases the value of the variable by 1	A++ = 8
–(Decrement)	decreases the value of the variable by 1	A- - = 6

The second type of operators are bitwise operators.Bitwise operators are operators that is applied to integer types, it performs bit by bit operations. The bitwise operators are:

Bitwise operators		
Operator	Description	Example
`& (bitwise and)`	Binary and operator	A & B = 3 (0111 & 0011 = 0011)
`| (bitwise or)`	Binary or operator	A | B = 7 (0111 | 0011 = 0111)
`~(bitwise xor)`	Multiplies the values on either side of the operator	A * B = 21
`^(bitwise compliment)`	Divides the left value from the right value	A / B = 2.3
`<<(left shift)`	Divides the left value from the right value then returns the remainder	A % B = 1
`>>(right shift)`	Increases the value of the variable by 1	A++ = 8
`>>>(zero fill right shift)`	decreases the value of the variable by 1	A- - = 6

Relational operators only return either true or false by comparing two variables. In the examples below A=5 and B=7 These are:

Relational operators		
Operator	Description	Example
==(equal to)	Checks if the values are the same on either side of the operator	A == B = false
!=(not equal to)	Checks if the values are different on either side of the operator	A != B = true
>(greater than)	checks if the left value is greater than the right	A > B = false
<(less than)	checks if the left values is less than the right	A < B = true
>=(greater than or equal to)	checks if the value of the left value is greater than or equal to the right	A >= B = false
<=(less than or equal to)	checks if the value of the left value is less than or equal to the right value	A <= B = true

There are also 3 logical operators in java, these also return true or false. In these examples A is True and B is False:

Logical operators		
Operator	Description	Example
`&&(Logical and)`	Checks if both variables are true/ non zero then returns true	`A && B = false`
||(Logical or)	Checks if one of the variables are true, then returns true	A || B = true
!(Logical not)	Used to reverse the logical state of its operand.	`!(A && B) = true`

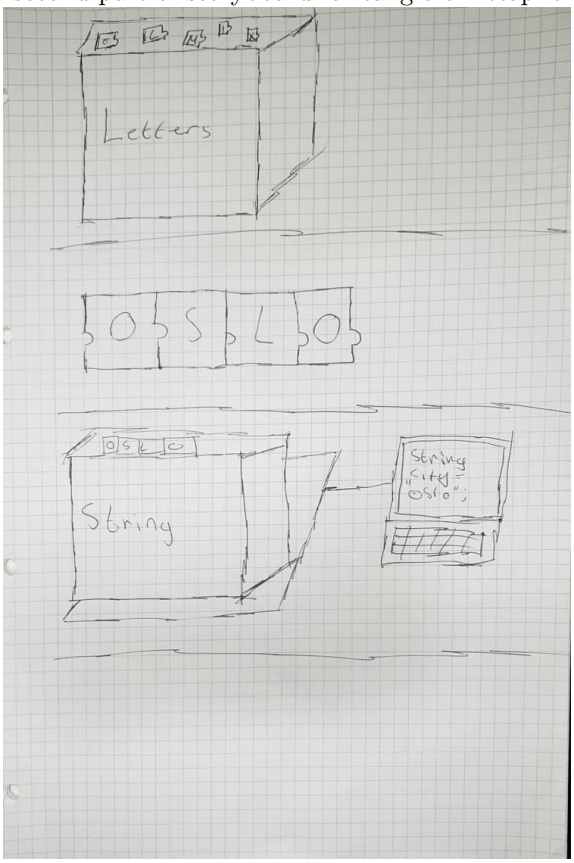
The last type of operators in used in java programming is called assignment operators. These are used to assign a value to a new variable. These are:

Assignment operators		
Operator	Description	Example
=	Assigns the value from the right side to the left side variable.	C = 5
+=	Adds the right side value to the left side value then assigns it to the left side variable	C += A same as C = C + A
-=	Subtracts the right side value to the left side value then assigns it to the left side variable	C -= A same as C = C - A
* =	Multiplies the right side value to the left side value then assigns it to the left side variable.	C *= A same as C = C * A
/ =	Divides the left side value with the right side value then assigns it to the left variable.	C /= A same as C = C / A
%=	Modulus and Assignment operator	C %= A same as C = C % A
`<<=`	Left shift and assignment operator	`C <<= 2 same as C = C << 2`
`>>=`	Right shift and assignment operator	`C >>= 2 same as C = C >> 2`
&=	Bitwise AND assignment operator	C &= 2 same as C = C & 2
`^=`	Bitwise exlusive OR and assignment operator	`C ^= 2 same as C = C ^ 2`
|=	Bitwise inclusive OR and assinment operator	C |= 2 same as C = C | 2

4.3.3 String and int

Because of the time limit of this master thesis I choose to first try making metaphors for a primitive and a non primitive variable, the data types I choose to focus on was a int and a String. First I made a storyboard for the string metaphor, as seen on the figure below.

Figure 4.4: second part of storyboard for tangible metaphor for string.



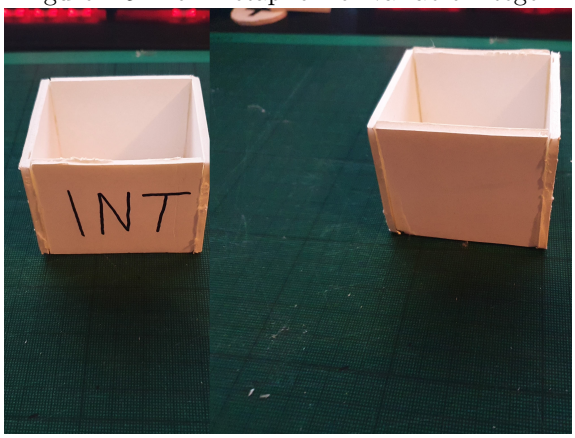
4.3.4 Paper prototype

After making the storyboards for the variables int and String, I choose to make a paper prototype to better illustrate my concept of how it will all work.

The box - variable

I choose to use model making foam paper to make the prototype since I was going to make a couple of boxes.

Figure 4.5: Box metaphor for variable integer.



The first box I made as seen in figure 3.6 was for the variable int, to distinguish what datatype the variable is, I choose to write it on the box.

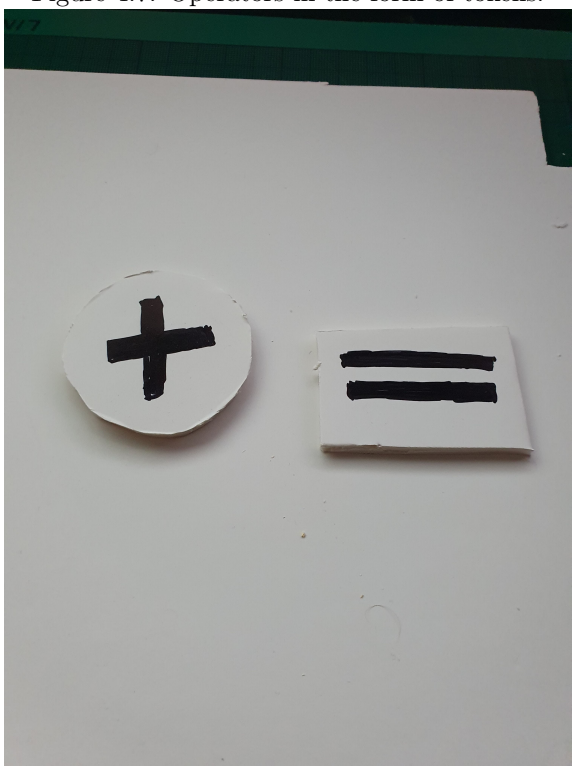
Figure 4.6: Box metaphor with top for variable integer.

Each box has a lid on the top symbolizing that you cant see what is inside the variable without a method to print the variable. The lid on the box is made out of the same foam paper, and the pivoting is made by poking a wooden dowel through both the box and the lid, as seen in figure 3.7.

Tokens - operators and input

To be able to use more than one variable together, I choose to add operators in the form of tokens, as seen on figure 3.8.

Figure 4.7: Operators in the form of tokens.



I also choose to use tokens as the input in variables, as seen on figure 3.9. Each token with input numbers fits into the variable box.

4.4 Prototype technology

The prototype has to be able to recognize variable boxes with numbers inside and it has to be able to recognize operators. It also has to be able to recognize when the user lifts the lid to the output box. This can be done by using different technologies. My plan is to use rfid technology for recognizing boxes with numbers inside and operators. A different approach could be using a camera then using image recognition for the boxes and operator, but that would then be much more complicated and would require much more setup by the user than using rfid which would just be plug and play. The plan is that the product will be used in teaching introductory programming so it has to be easy to set up and easy to pack up and carry to the next class room, or else there is a bigger change of it not getting used, that is why I want to use rfid technologies to try making a tangible metaphor toolkit. The technologies need to be able then to recognize boxes placed on the table and operators, for the variable prototype it needs to be able to recognize two boxes, one operator and the user lifting the lid of the result box.

The problem with a arduino is that it can only have one nfc shield, see figure 4.8

connected to it, so it can only read nfc tags on one location. This problem I plan on solving with using three arduinos with three nfc shield. Arduinos does not support threads, so when it is listening to the nfc shield it can not do anything else like talk to the other arduinos and display information. This problem I plan on solving with using a raspberry pi, see figure 4.9 which can get all the information from the arduinos while also displaying the corresponding information. The raspberry pi will also be responsible for checking if the user has lifted the lid of the result box. I plan on solving this by using some copper wire and copper tape, when the raspberry pi reads that the signal is broken it will display the result of the operation done by the user.

Figure 4.8: Arduino nfc shield.

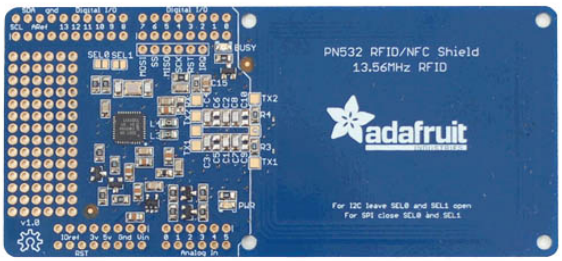
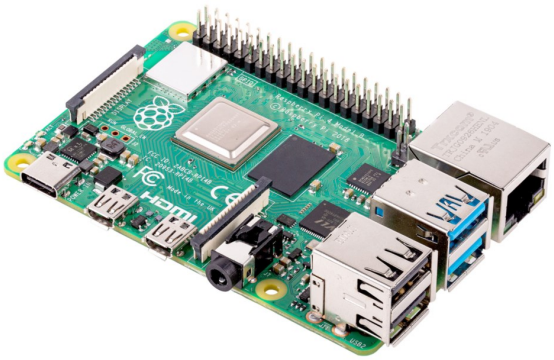


Figure 4.9: Raspberry pi.



4.4.1 Parts

For the prototype I plan on using three arduinos with nfc shield that are all connected to a raspberry pi that will control them and display the corresponding code. Two of the arduinos will be used to check which operators gets placed and the third arduino will be used in one of the boxes, to check which number the user places inside.

4.4.2 Arduino - RFID testing

I used three sparkfun arduinos as seen on figure 4.1, with three RFID-RC522 rfid shields as seen on figure 4.2.

Figure 4.10: Arduino

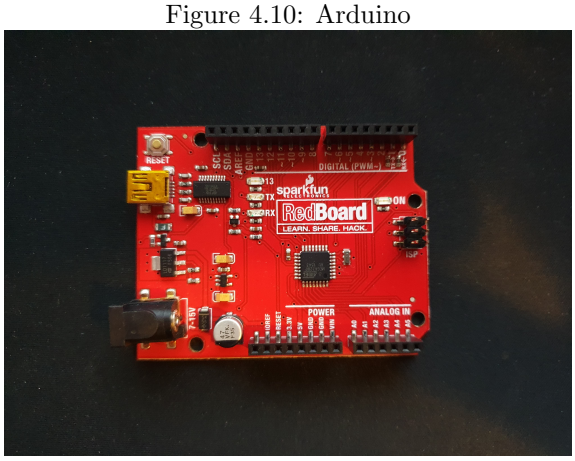


Figure 4.11: RFID shield

I then used a breadboard to connect the rfid shield to the arduinos with wire as seen on figure 4.3. The first test I ran was on the rfid shield. I tried connecting it to the arduino and then reading a rfid chip, as seen on figure 4.4. When the rfid chip is placed near the rfid shield the arduino scans it and outputs the UID tag, which is the id of the chip. After scanning two chips I changed the code to output one chip as the + operator and the other as the = operator when scanned, as seen on figure 4.5.

Figure 4.12: Arduino rfid test

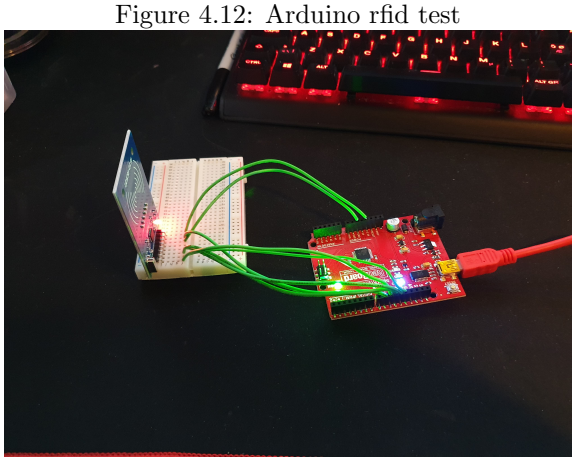


Figure 4.13: output from the first rfid test.

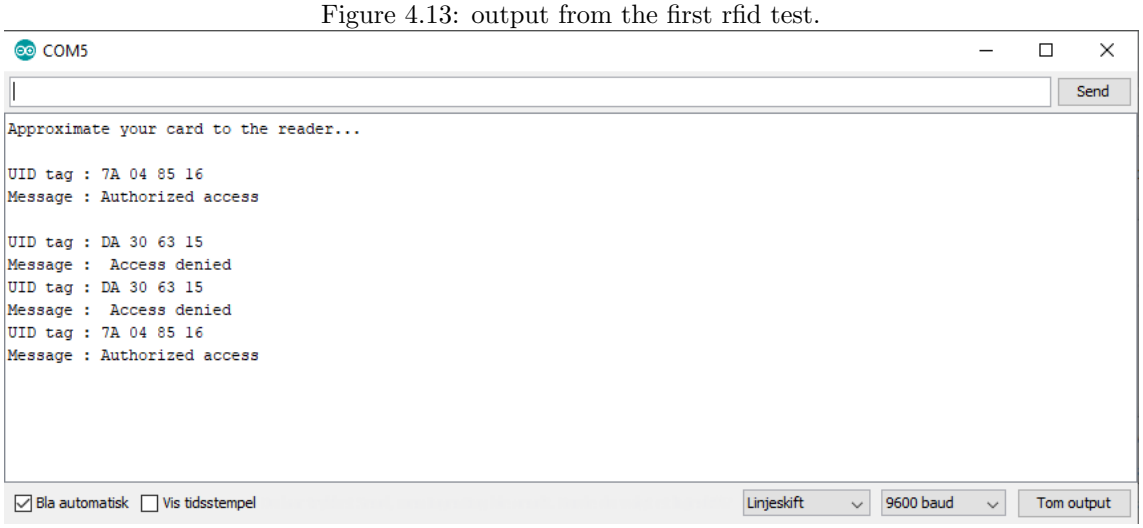


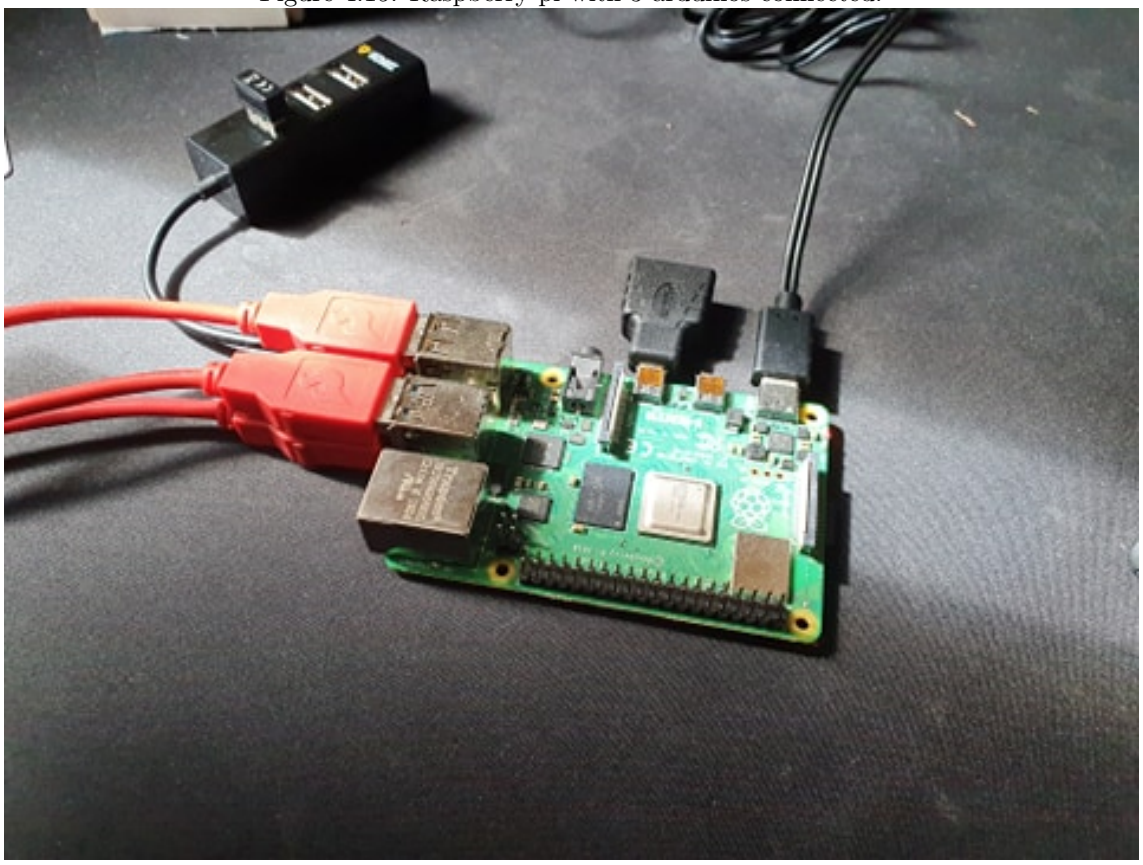
Figure 4.14: output from the second rfid test

```
Approximate your card to the reader...

UID tag : DA 30 63 15
Message : You have placed = as the operator
UID tag : 7A 04 85 16
Message : You have palced + as the operator

UID tag : 9A A6 57 0F
Message : You have palced a invalid operator
```

4.4.3 Raspberry pi

For my project I will be using a raspberry pi 4 computer running raspbian. The raspberry pi will be getting the information from all three arduinos, as seen on figure 4.20. It is running python code, which listens to the usb-inputs from the arduinos, then when there has been placed one nfc tag on the reader the ardunios will send that id to the raspberry pi. The code then checks which value the variable has and which operator has been placed on the reader. When a variable has been placed on the reader with a value the code prompts the user to write the name of the variable on a keyboard connected to the raspberry pi. When the user has placed and named all the variables the movments of the boxes will be translated to java code and displayed on the screen. Lastly the user can pick up the lid of the box on the end to get the output from the variables.

Figure 4.15: Raspberry pi with 3 arduinos connected.



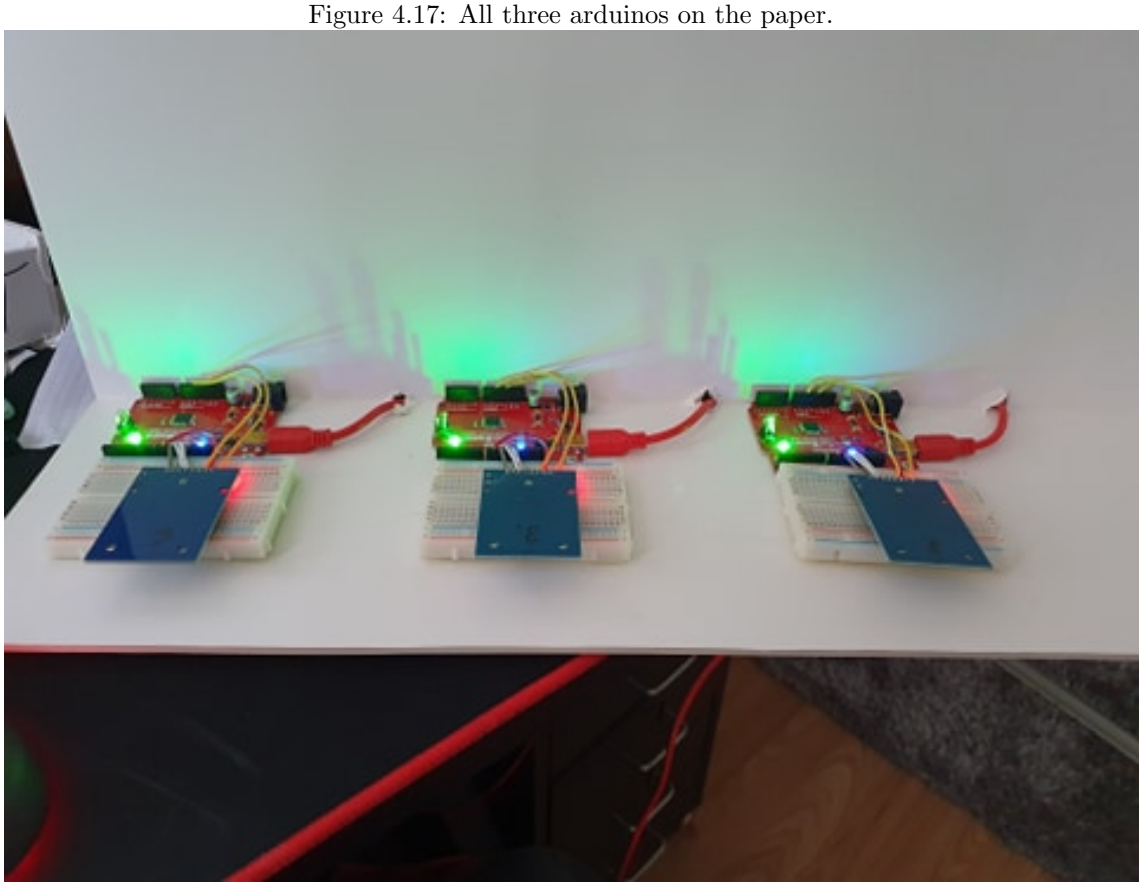
4.4.4 High fidelity Prototype

For the prototype I chose to use the same token and boxes I made for the paper prototype. I started by just putting nfc tags on the underside of all the tokens and numbers, as seen on figure 4.21.

Figure 4.16: Tokens with nfc tags on.

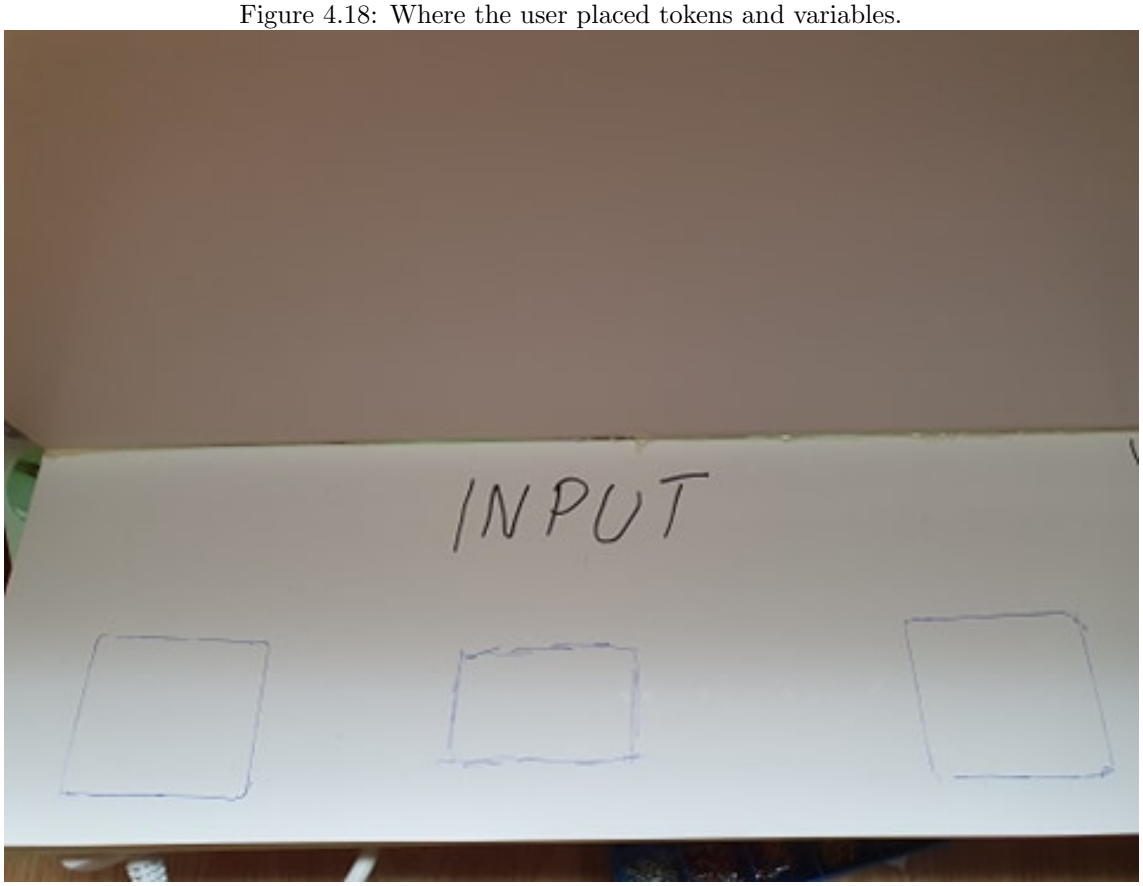
Then I used a big piece of the architect paper. to glue the three ardunios with the nfc shield on the bottom, as seen on figure 4.22

Figure 4.17: All three arduinos on the paper.



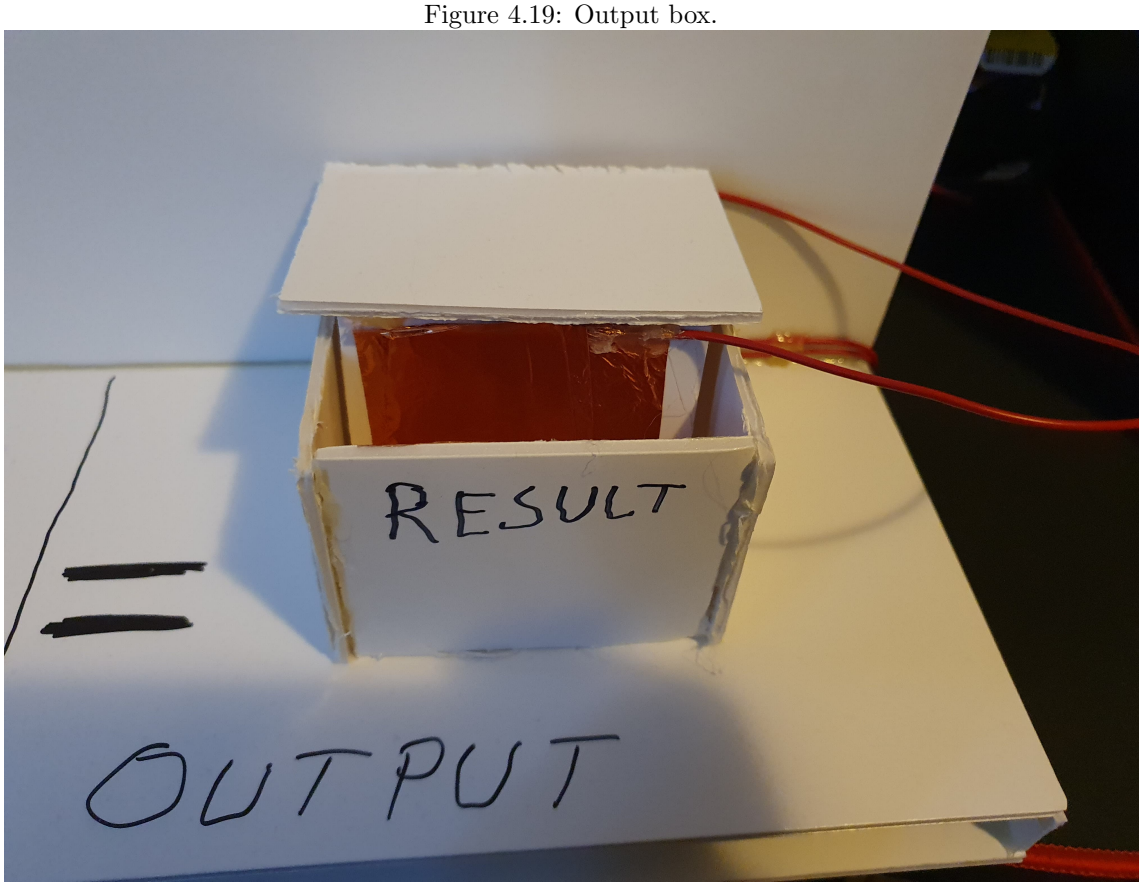
For covering them up and making it prettier I used another piece of the paper on top and glued the sides on so it stays in place, as seen on figure 4.23, and marked where each nfc reader is so that the user knows where to place the tokens and variable boxes.

Figure 4.18: Where the user placed tokens and variables.



I then used the last bit of space on the paper to make one more box with a lid on top, as seen on figure 4.24 The lid is used as a metaphor for printing out the output. Lid as a metaphor for printing out the output did not come from the literature review. This is something I came up with as a tangible task that the user can do to represent the action of a print code. When the user lifts the lid of the box the circuit breaks and the arduino send a signal to the raspberry pi, that then prints out the code. For making the connection in the box, I just ran two wires from one of the arduinos and connected one of them to the box with using copper tape, and the other one to the lid the same way.

Figure 4.19: Output box.



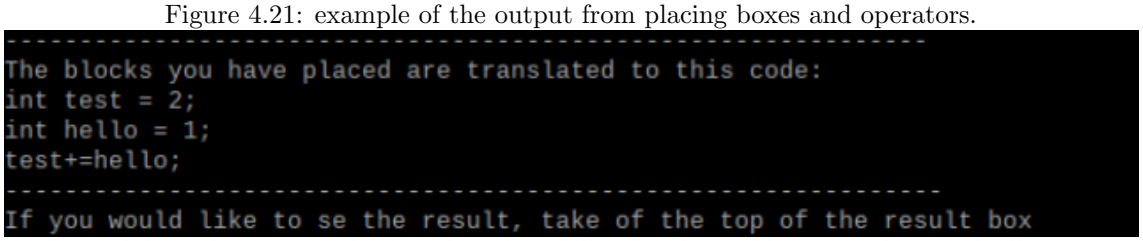
Placing variables

When program start it prompts the user to place a box with a number inside on the first square. By placing the box with a number inside, it triggers the raspberry pi to prompt the user to choose a name for the variable, as seen on figure 4.25. When the user places the operator the program just writes which operator the user has placed. And when the user places the last variable the program prompt the user for a name, same as the first time. When the user has placed all the variables and operators the program translates the placements of the boxes and variables to java code and prints them out, as seen on figure 4.26

Figure 4.20: variable name prompt

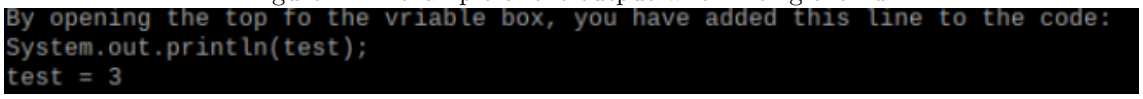
```
Start by placing a variable box with a number inside on the first square
You have placed a int variable with the value 2
Write the name of the variable:test
```

Figure 4.21: example of the output from placing boxes and operators.



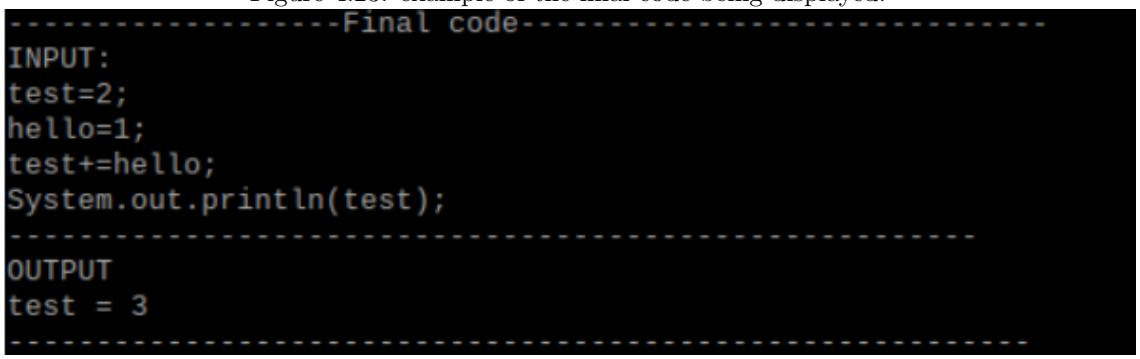
When the user opens the lid of the output box the programs writes out that the user has asked for the output from the operation done previously, as seen on figure 4.27

Figure 4.22: example of the output when lifting the lid.



After 3 seconds from when the user lifts the lid the final code with split input and output will be displayed, and the user can start over, as seen on figure 4.28.

Figure 4.23: example of the final code being displayed.



4.5 User testing

The user testing and interviews was planned to be done with students at the campus in Høgskolen I Østfold Remmen. The target group for this project was students that study programming. But under the circumstances of the pandemic in the world, I could not conduct any user testing on the target group. Based on the restrictions in Norway I could only do the user testing on those people that live in the same house as me, and some close friends. The plan was to have at least around 20 users try the product, but I had only acess to 7 people that could try the prototype and answer the interview questions.

4.5.1 Test plan

Since I could not do the user testing on campus with the intended target group, I had to do conduct the testing at home. Each user got to try the prototype for as long as they wanted, after they said that they were done with the prototype I conducted a semi structured interview with each user. The questions I had planned on asking them are discussed in the sub chapter questions.

4.5.2 Questions

The questions I had planned for the semi structured interview was designed to answer the proposed research questions in my thesis. The first question is meant to see if the user has any programming knowledge before trying the prototype. The second question was meant to find out if the user knows what a variable is before trying the prototype, Maybe some users knew some programming but did not know what a variable is. The third question is to see if the user knows what a variable is after testing the prototype, if they answer yes they had to describe it. Question four is meant to get the users feedback on using the box as a metaphor for variable, and the fifth question is to get the feedback on using a lid as a metaphor for the results. The last question is meant to see if the user thinks anything has to be changed in the prototype.

1. Do you have any experience with programming?
   -Yes
   -No

2.Before trying the prototype, did you know what a variable was?
   -Yes
   -No

3.After using this prototype, did you understand what a variable was? if yes, could you describe it?
   -Yes, and explanation.
   -No

If a person understood what a variable was then I would ask:

4.Was using the box a appropriate metaphor for a variable?
   -Yes
   -No

5.Was the lid a appropriate metaphor for seeing the result of the operation?
   -Yes
   -No

6.Was there anything you would like to change about the prototype or the metaphors

used?
Then the last question would be, what the student though about using this as a supplementation to the standard teaching methods used today in the classroom, and if they would see any benefit to implementing it. Since I could not do the user testing and interviews on the planned target group that question had no effect.

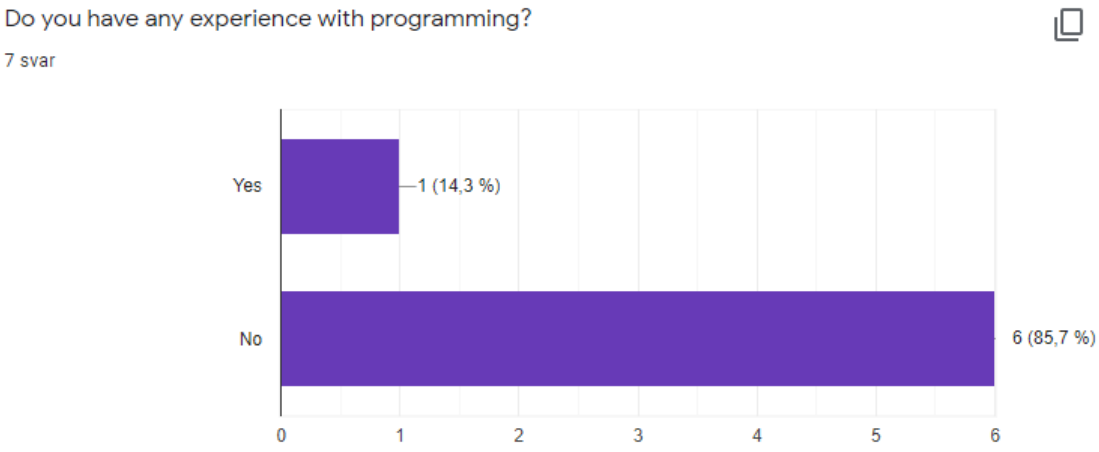
4.5.3 Results

As stated before the user testing could not be conducted as it was planned based on the restrictions that the Norwegian government placed on socializing. That meant that the school I had planned on conducting the field test on and the interviews was empty and had no students present. That meant that I had to user the people that lived in the same house as I, and some close friends that I could get to try it out. It also meant that the user test had to have a lot fewer people testing it then if it could be conducted on the target group. Each Person first got to test the prototype for as long as they wanted, then I followed up with a interview based on the questions from the previous sub chapter.

Question one

The first question is meant to see if the person knows anything about programming before they started using the prototype. Since I could not test it with my planned target group I had to use the people around me. All but one of the people I had access to did not have any knowledge about programming before they started using the prototype, as seen on figure 4.24. This is good since my target group are students that have no experience in programming or have just started learning.

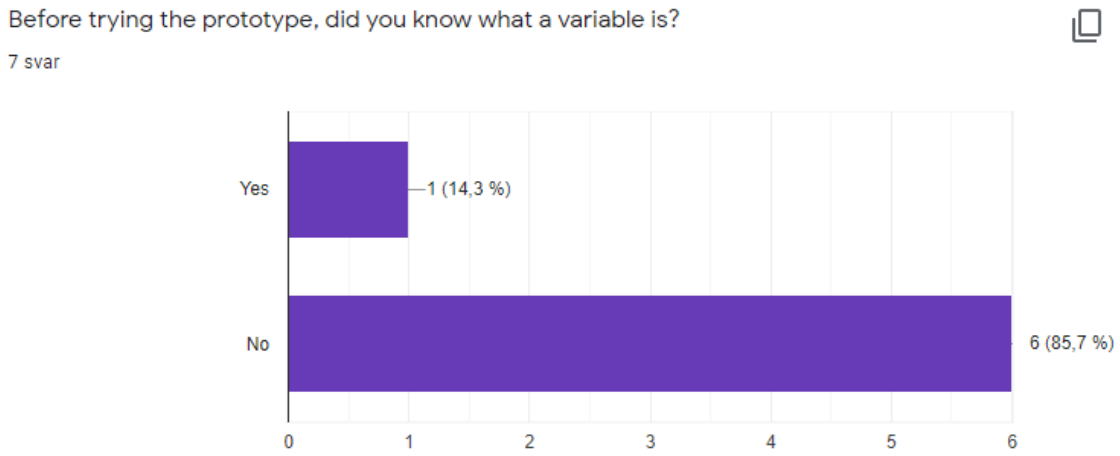
Figure 4.24: Graph of the answers from question one.



Question two

Question two was meant to see if they knew what a variable is before trying the prototype. All but the one person who had experience from programming before did not know what a variable was, as seen on figure 4.25. Based on that its easier to see if they learned anything by using this approach to learning programming concepts.

Figure 4.25: Graph of the answers from question two.



Question three

The third question is designed to find out if the user understood what a variable is after using the prototype, if the user answers yes to understanding the question I ask them to describe what a variable is. Here all but one answered yes to understanding what a variable is, see figure 4.26. Just one person did not understand what a variable is after testing. The six users that answered yes also gave a good description of what a variable is and what you can use it for, see figure 4.27. Based on the descriptions that the users gave of what a variable was, I would say that the prototype conveyed the basics of what a variable is and what it can be used for.

Figure 4.26: Graph of the answers from question three.

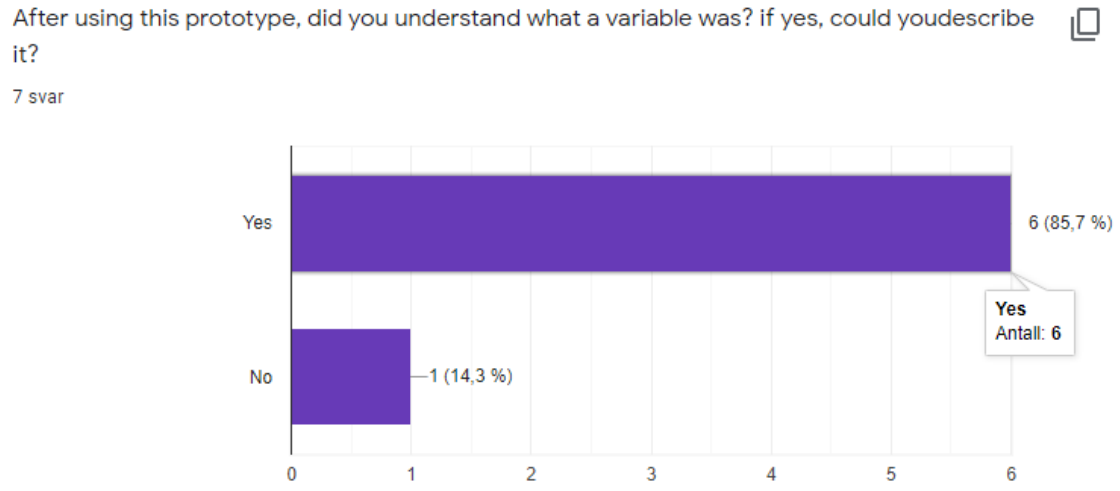
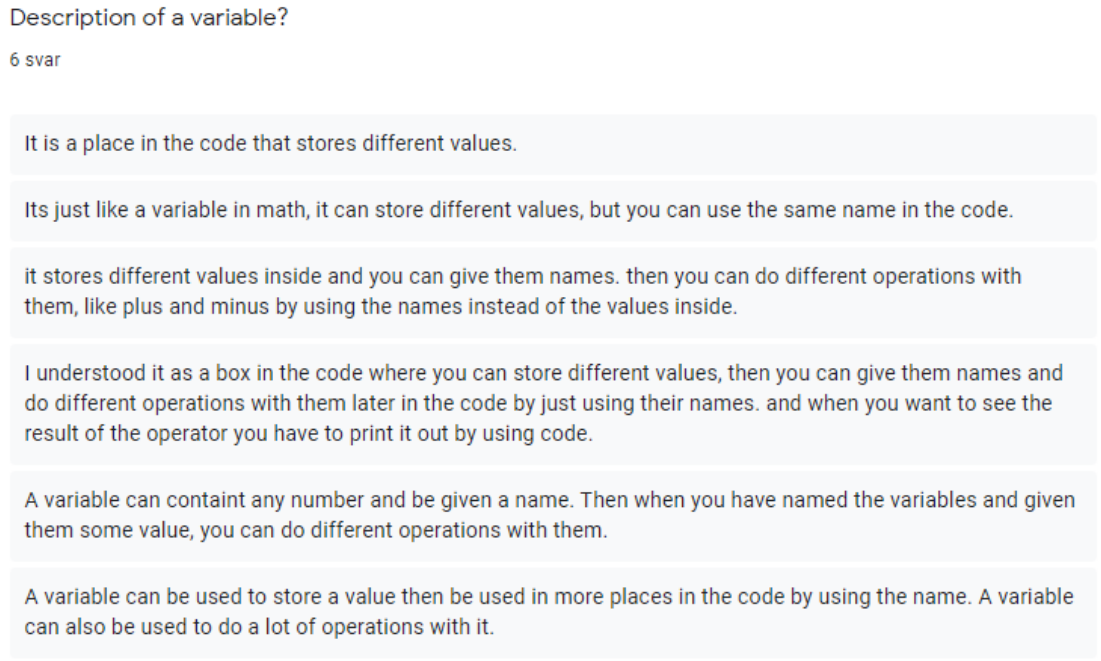


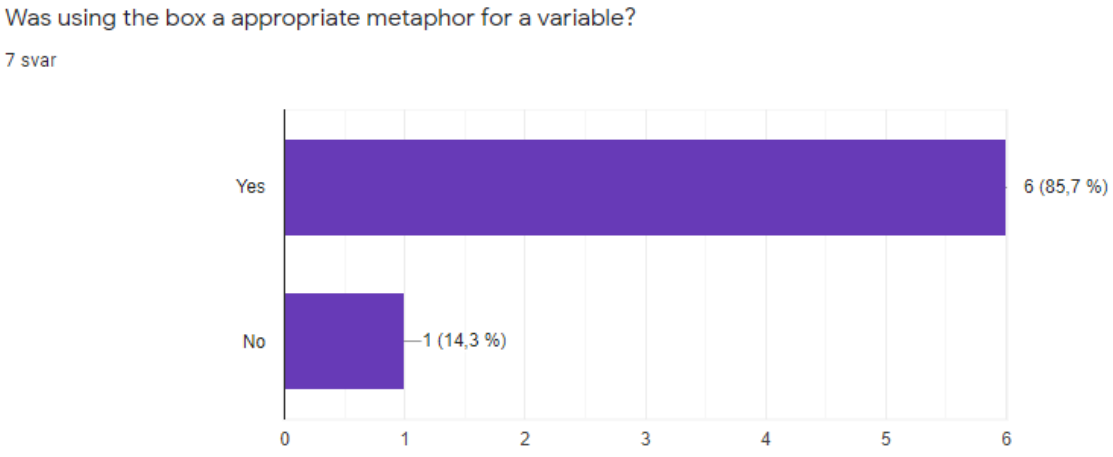
Figure 4.27: Graph of the answers from question four.



Question four

Question four is made to get the users perspective on using a box as a metaphor for variable. All but the one person who did not understand what a variable was though that a box is a good metaphor for variable, see figure 4.28.

Figure 4.28: Graph of the answers from question five.

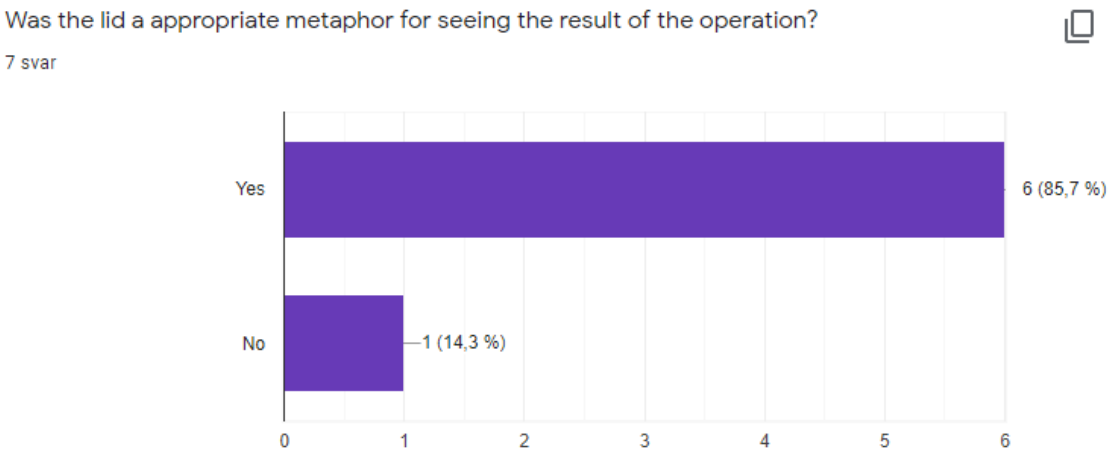


Question five

Question five is also made to get the users perspective on using a box with a lid as a metaphor for printing out the result. In this question also all but the one user who did

not understand what a variable was after testing the prototype though that it was a good metaphor, see figure 4.29. All who answered yes to this question understood that the code will not show the result from operations without having some form of code to print it out.

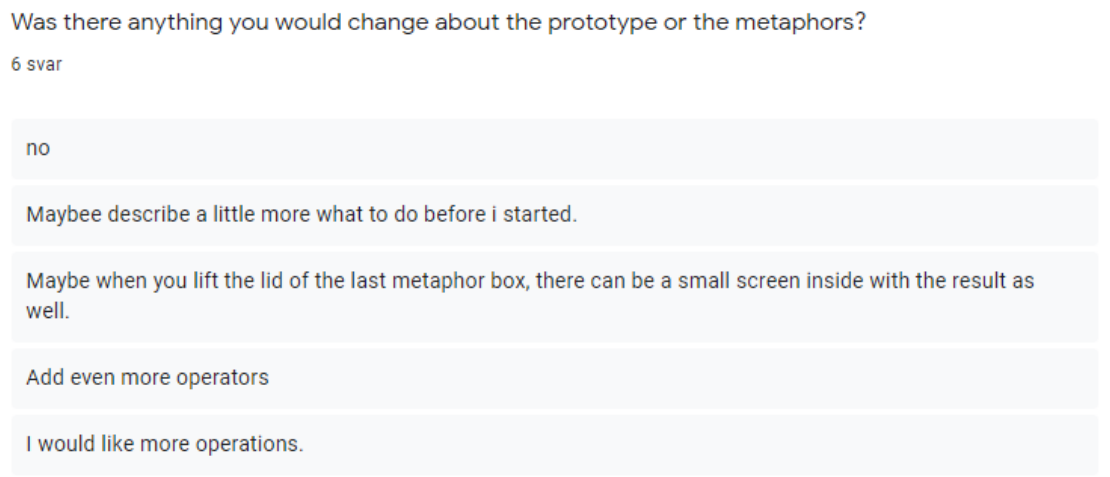
Figure 4.29: Graph of the answers from question six.



Question six

The last question was designed to get feedback from the users if there was anything that they would like to change with the prototype, see figure 4.30 for all answers. One of the users said that they needed more information before starting, They though that it was not clear enough. One good feedback was to add a small screen in the result box that will also show the result of the operation done.

Figure 4.30: Graph of the answers from question one.



Chapter 5

Discussion

Metaphors have been used previously in teaching in general, and tangible toolkits have been used to teach programming to children. Both aspects have shown positive learning effects, which indicate that metaphors and tangible toolkits can be conjoined as a new method to teach students programming. The tangible metaphor toolkit can be used as a supplement to the standard curriculum to teach students programming concepts.

Based on the user test that was conducted, the metaphor based tangible programming toolkit shows promising results. All but one of the user did not have any experience in programming or any knowledge about programming before they started using the prototype. After each user tested the prototype, all but one person could describe the main essence of a variable, and what they can be used for. It is the same for the questions asking if a box is a appropriate metaphor for a variable, and if the lid was a appropriate metaphor for printing the result. All but one person though that it was appropriate, the only person that though that it was not appropriate was the one person that still did not understand what a variable was after using the prototype. Using lid as a metaphor for seeing the result of the operation done by the user with the variables was something I came up with in the prototype and has not taken out from the related work. This metaphor did have positive outcome in the user test. One thing that one of the users suggested which would have been a good idea to add to the prototype was to add a little screen or some form of display inside the result box. This would then also display what the result was from the operation inside the box, and not only show up on screen.

The concept for this project was to make a metaphor based tangible toolkit for programming education. So it was important for the prototype to be easy to set up. I choose to use three arduinos and one raspberry pi. The arduinos are all wired to the raspberry pi, so the user only has to connect a monitor and plug it into the wall socket to be ready to use the prototype. The prototype would also be able to be build by using a camera and image recognition, but this would be harder for the users to set up so that the camera could get a clear picture of all the boxes and operators.

The prototyping process answers the first research question which was: "How can I make a tangible metaphor to describe a programming concept." The metaphor for a variable is taken directly out from previous research on metaphors for programming concepts. A box has been used as a metaphor in many studies done previously, as discussed in the chapter Related work. But the metaphor for seeing the result, which I choose to use a lid on a box. That meant that the user had to open the box to see the result done by the operations. This metaphor was well received in the user test done at the end. The second

research question is harder to measure, based on the restrictions placed on socializing in Norway by the government, which meant that I could not do the user testing on the planned target group. That meant that the users had to be the people I had access to around me. Those were family that lived in the same house as me and some close friends. I conducted a user test where the user go to try the prototype for as long as they liked, then I conducted a semi structured interview with each user after they were finished. By doing the semi structured interviews I had the freedom to go outside the questions I had planned before the interviews. The results gotten from the user tests were positive and they could describe what a variable was, but it still could have some bias on the last questions if the box and lid is appropriate metaphors for the programming concepts. Also I would have liked to do the user testing on a much larger group of people and on the planned target group for this project. /

Chapter 6

Conclusion and future work

Metaphors and tangible programming have been used separately in teaching students previously. Both have shown positive learning effects, and combining these two as a new method or tool for teaching programming. The tangible metaphor based programming toolkit shows positive learning effects based on the user test that was conducted. Because of the restrictions placed by the Norwegian government I could not conduct the user test as planned in the start of the project, but I had to do a smaller user test not on the target group. A bigger user test and interviews could be run on the target group as future work. The prototype shows positive learning effects on the users I had access to, based on the results from the user testing and interviews conducted at the end of each test.

For future work on the prototype, one of the users in the user test proposed adding a screen inside the result box which could be a good way to illustrate the output on the prototype as well as on the screen. Also before running a bigger test on the planned target group, it could be better to have one more tangible metaphor for a programming concepts, so that the students could try more than just variables.

Since I could not do the planned field study on the proposed target group, a good way to continue the research would be doing a much bigger field study with interviews after on the proposed target group. I had planned on asking if they thought that using this toolkit would be beneficial to the standard curriculum, but since I could not do the tests on the target group that question fell thru, because none of the participants had any previous knowledge of programming or programming education. And by doing a field study on the proposed target group the bias of that all the participants had close relationships with me.

Bibliography

[Chawla et al., 2013] Chawla, K., Chiou, M., Sandes, A., and Blikstein, P. (2013). Dr. Wagon: A 'stretchable' toolkit for tangible computer programming. In *ACM International Conference Proceeding Series*, pages 561–564.

[Gassner, 1999] Gassner, G. J. (1999). Using Metaphors for High-Performance Teaching and Coaching. *Journal of Physical Education, Recreation & Dance*, 70(7):33–35.

[Gordon et al., 2015] Gordon, M., Rivera, E., Ackermann, E., and Breazeal, C. (2015). Designing a relational social robot toolkit for preschool children to explore computational concepts. In *Proceedings of IDC 2015: The 14th International Conference on Interaction Design and Children*, pages 355–358.

[Hidalgo-Céspedes et al., 2018] Hidalgo-Céspedes, J., Marín-Raventós, G., Lara-Villagrán, V., and Villalobos-Fernández, L. (2018). Effects of oral metaphors and allegories on programming problem solving. *Computer Applications in Engineering Education*, 26(4):852–871.

[King, 1997] King, K. N. (1997). The case for Java as a first language. *Proceedings - 35th Annual Southeast Regional Conference, ACM-SE 1997*, pages 124–131.

[Kruglyk and Lvov, 2012] Kruglyk, V. and Lvov, M. (2012). Choosing the First Educational Programming Language. Technical report.

[Lynch and Fisher-Ari, 2018] Lynch, H. L. and Fisher-Ari, T. R. (2018). What We Learned about Using Metaphors in College Teaching: Methods and Meanings.

[Mckay, 1999] Mckay, E. (1999). Exploring the Effect of Graphical Metaphors on the Performance of Learning Computer Programming Concepts in Adult Learners: a pilot study. *Educational Psychology*, 19(4):471–487.

[McLeod, 2014] McLeod, S. (2014). The Interview Research Method — Simply Psychology.

[McLeod, 2015] McLeod, S. (2015). How to Conduct User Observations — Interaction Design Foundation.

[Mortensen,] Mortensen, D. H. User Research: What It Is and Why You Should Do It — Interaction Design Foundation.

[Pérez-Marín et al., 2018] Pérez-Marín, D., Hijón-Neira, R., and Martín-Lope, M. (2018). A Methodology Proposal Based on Metaphors to Teach Programming to Children. *Revista Iberoamericana de Tecnologias del Aprendizaje*, 13(1):46–53.

[Rikke Friis Dam,] Rikke Friis Dam, T. Y. S. Design Thinking: Get Started with Prototyping — Interaction Design Foundation.

[Robins et al., 2003] Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. *International Journal of Phytoremediation*, 21(1):137–172.

[Rogozhkina and Kushnirenko, 2011] Rogozhkina, I. and Kushnirenko, A. (2011). PiktoMir: Teaching programming concepts to preschoolers with anew tutorial environment. In *Procedia - Social and Behavioral Sciences*, volume 28, pages 601–605.

[Sajaniemi et al., 2007] Sajaniemi, J., Byckling, P., and Gerdt, P. (2007). Animation Metaphors for Object-Oriented Concepts. *Electronic Notes in Theoretical Computer Science*, 178:15–22.

[Salleh et al., 2013] Salleh, S. M., Shukur, Z., and Judi, H. M. (2013). Analysis of Research in Programming Teaching Tools: An Initial Review. *Procedia - Social and Behavioral Sciences*, 103:127–135.

[Sapounidis et al., 2019] Sapounidis, T., Demetriadis, S., Papadopoulos, P. M., and Stamovlasis, D. (2019). Tangible and graphical programming with experienced children: A mixed methods analysis. *International Journal of Child-Computer Interaction*, 19:67–78.

[STAPPERS and GIACCARDI,] STAPPERS, P. and GIACCARDI, E. *The Encyclopedia of Human-Computer Interaction, 2nd Ed.*, chapter 43.

[Truong et al., 2006] Truong, K. N., Hayes, G. R., and Abowd, G. D. (2006). Storyboarding: An Empirical Determination of Best Practices and Effective Guidelines.

[Vujošević-Janičić and Tošić, 2008] Vujošević-Janičić, M. and Tošić, D. (2008). The role of programming paradigms in the first programming courses. *Teaching of Mathematics*, 11(2):63–83.

[Waguespack, 1989] Waguespack, L. J. (1989). Visual metaphors for teaching programming concepts. *ACM SIGCSE Bulletin*, 21(1):141–145.

[Wang et al., 2013] Wang, D., Zhang, Y., and Chen, S. (2013). E-block: A tangible programming tool with graphical blocks. *Mathematical Problems in Engineering*, 2013.

[Zimmerman et al., 2007] Zimmerman, J., Forlizzi, J., and Evenson, S. (2007). Research Through Design as a Method for Interaction Design Research in HCI.