2017

Tangible programming languages a better solution to teaching children to program?

Student Name: John Wilkie
Candidate Number: 118499
Degree: Computer Science
School: Informatics
Supervisor: Dr. Judith Good
Word count: 11997

Statement of originality

This report is submitted as part of the requirement for the degree of Computer Science at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Signed:

Date:

Acknowledgements

I would like to thank my main supervisor, Judith Good, for the guidance and support that was generously given while I worked on this project and report. I would also like to thank Peter Cheng for his input during his time supervising the project. I must also thank all the participants that gave me valuable feedback during the testing and evaluation sessions.

Summary

The aim of this project was to build a system to explore the differences between tangible and screen based interfaces when teaching children aged 5 – 7 years old how to program. The system created involved building a new tangible programming language and a screen based equivalent, an Android app to evaluate the programming language and record data, and a physical robot that could be programmed using the new language. This report explores the design and implementation process of the system components and some key issues encountered.

A small study was carried out involving 14 adults using the system. Although a larger scale study with the target population is needed, preliminary results indicated that users who used the tangible interface reported a higher level of enjoyment than those using the screen interface. In contrast, those using the screen interface would on average complete a problem quicker than those using the tangible interface. One outcome of this practical study was that it highlighted a number of improvements which could be made to the system and uncovered some interesting research questions for future study.

Contents

Statement of originality 2
Acknowledgements 3
Summary 4
1. Introduction 8
1.1 Report overview 8
1.2 Project introduction 8
1.3 Project objectives 8
2. Background Literature 10
2.1 User research and considerations 10
2.2 Review of previous studies 10
2.3 Review of teaching children to program and tangible programing languages 11
3. Professional considerations 15
3.1 Project compliance with BCS Code of Conduct 15
3.1.1 Public interest 15
3.1.2 Professional Competence and Integrity 15
3.2 Ethical considerations and ethical compliance 15
4. Requirements analysis 16
4.1 Tangible block requirements 16
4.2 App requirements 16
4.3 Robot requirements 17
5. System Design and Implementation 19
5.1 Choice of technology for system components 19
5.2 App design 20
5.2.1 App architecture 20
5.2.2 Use case diagram 20
5.2.3 Use case scenarios 21
5.2.4 Activity diagrams 21
5.2.5 Logic tier 23
5.2.6 Data layer 25
5.2.7 Presentation tier 26
5.3 Tangible design 29
5.3.1 Initial tangible prototypes 29
5.3.2 Tablet holder and mirror design 33
5.4 Robot design 34
5.4.1 Initial robot design 34
5.4.2 Initial robot prototype 35
5.4.3 Tiles 39

6. System implementation 41
6.1 App development 41
6.1.1 Test driven development 41
6.1.2 Version control 41
6.1.3 Code style 41
6.1.4 Extending android classes 41
6.1.5 Multithreading 42
6.1.6 Transferring and sharing data between activities 43
6.1.7 Final app screens 44
6.2 Tangible Implementation 45
6.3 Robot implementation 48
6.4 Overall system 50
7. System testing and Evaluation techniques 52
7.1 System testing 52
7.2 Gold standard evaluation 52
7.3 System evaluation 52
7.3.1 Aim of testing 52
Participant role 52
Facilitator role 53
7.3.2 Participants 53
Participant role 53
Facilitator role 53
7.3.3 Materials used 53
Participant role 53
Facilitator role 53
7.3.4 Testing procedure 53
Participant role 53
Facilitator role 54
7.3.5 Evaluation problems 54
8. Evaluation findings 55
8.1 Participant role findings 55
8.2 Facilitator role findings 58
8.3 Future improvements 59
8.3.1 Robot improvements 59
8.3.2 Language improvements 60
8.3.3 App improvements 61
8.3.4 Stand and tile improvements 61
9. Conclusion 63

9.1 Project conclusion 63
9.2 Further study 63
10. References 64
Appendix A 67
Meeting log 67
Appendix B 68
Ethical compliance 68
Appendix C 70
Use case scenarios 70
Activity diagrams 72
Appendix D 75
Participant role documents 75
Introduction script 75
Pre-questionnaire 76
Tasks 77
Post-questionnaire 78
Debriefing script 79
Participant role results 80
Appendix E 83
Facilitator role documents 83
Introduction script 83
Tasks 84
Instruction sheet 85
Post-questionnaire 86
Debriefing script 86
Facilitator role results 87
Appendix F 88
Heuristic evaluation 88
Appendix G 90
Test plan 90
Appendix H 95
Libraries used 95
Android libraries 95
Arduino libraries 95
Appendix I 95
Ethical consideration 95
Draft ethical application and supporting documents 96

1. Introduction

1.1 Report overview

This report describes the aims of the project, explains the background reasoning and professional and ethical considerations required, then outlines the requirements set for the project to achieve. Details are given regarding the design work, implementation, testing, and finally a critical evaluation of the project is presented with resulting conclusions.

1.2 Project introduction

Globally there has been growing concern over failings in education systems to educate younger people not only to be safe consumers of digital technology, but also to understand how it works and to be able to be creators themselves [1]. In September 2013, the Department for Education in England made significant changes to the National Curriculum: what was previously known as "ICT" became "Computing Programs of Study" [2]. As a result, from key stage 1, children (aged 5 – 7 years) are required to be able to create and debug simple programs and be able to understand what algorithms are and how to implement them.

This has resulted in a large increase in the number of young children being taught to program, which has yielded increasing focus regarding the manner in which they are taught. One approach to teaching younger children to program has been to use tangible programming languages (TPLs). TPLs use physical objects that represent instructions that can be manipulated to write programs in a physical manner. Some people feel that these TPLs are a better solution for teaching young children to program than more traditional screen based systems. One such TPL, Quetzal [3], is shown in Figure 1.

Figure 1: Tangible programing language Quetzal [3]

However, little research has been conducted to assess the benefits (if any) of tangible programming over screen based equivalents. This project aims to explore these differences in particular by using a tablet computer for the screen based equivalent. Tablet computers are fast becoming the standard computing interface for children. According to a study run by Ofcom [4], one in every three children have their own tablet computer at home, and in 2015 BESA (British Educational Suppliers Association) [5] found that 71% of primary schools are now using tablets in classrooms (up from 56% in 2014). The majority of previous studies have used desktop computers to compare against the TPLs. The use of a mouse as input places control in the hands of a single child, whereas using a tablet allows multiple children to touch the screen at once. The difference between mouse-based and touch-based input is a potential topic of interest for many reasons, but is out of the scope of this study.

1.3 Project objectives

The overall aim of the project was to investigate the differences between a TPL and a screen based equivalent. So the first of two primary objectives was to create a simple TPL with a small number of instructions and a screen based equivalent (on a tablet computer) designed for key stage 1 children (5 – 7 years old). This was a non-trivial task involving multiple systems communicating with each other and working with unfamiliar technology. The systems used to implement the two interfaces had to be able to record data on the way that they were being interacted with.

The two interfaces created are used to program a physical robot to move around on a grid in order to solve a task. Building this robot, that must work in combination with either interface, was the second primary objective.

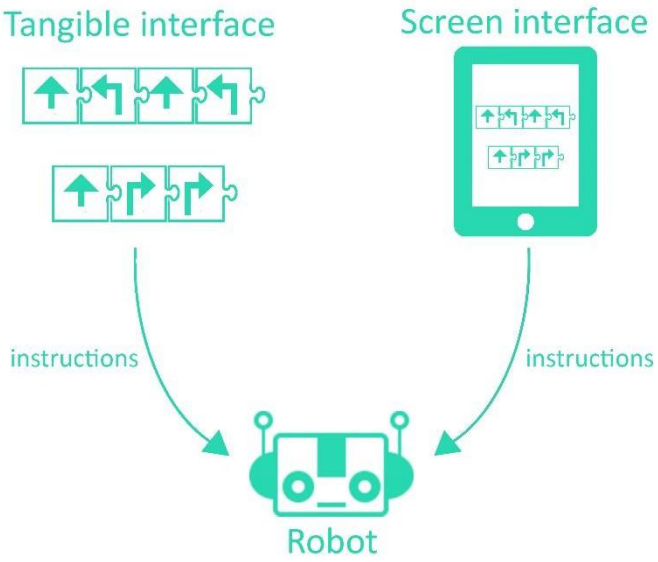

Figure 2: The two programming interfaces.

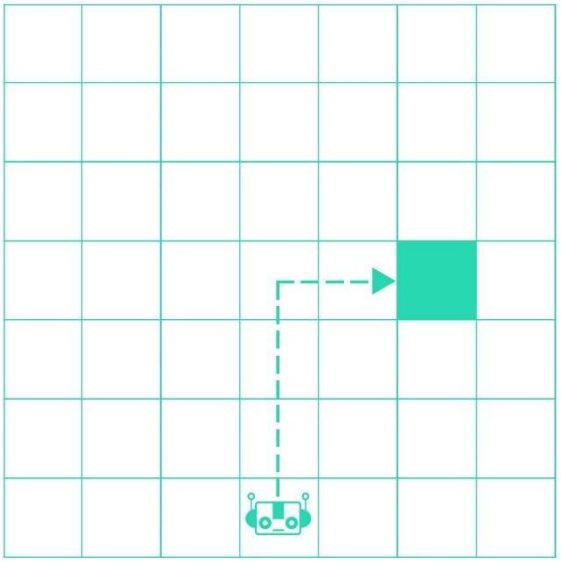

Figure 3: A simple task for the robot to complete.

The project also had three secondary objectives, intended to be implemented provided sufficient time was available.

1. Carry out a small scale study with key stage 1 children in a school environment. Gather data on how groups of children interact with the two interfaces while completing tasks with the robot.
2. Add more programming instructions to the initial simple language and build screen based equivalents. For example an 'if' statement that checks if the robot's path is obstructed and performs a given instruction if it is. If it was found that TPLs are of benefit to the target user group, these more complex instructions could be used in future studies to investigate if there is a point in program complexity where TPLs stop providing a benefit.
3. Create a hybrid interface that incorporates both tangible and screen based elements. The screen interface will update in real time to represent the tangible objects being manipulated in view of the camera.

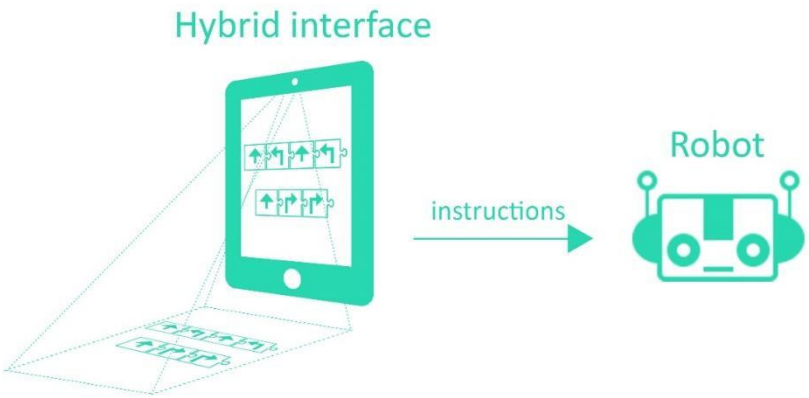

Figure 4: The hybrid interface.

2. Background Literature

This section of the report highlights the special considerations required for designing for the target user group, reviews previous studies in the use of TPLs and takes a brief look at the history of both teaching children to program and using TPLs.

2.1 User research and considerations

Ideally, during the research phase of this project, observations would have been taken of the target user group using a number of TPLs. However, due to limited access to TPLs and the need for high-risk ethical review, this was not feasible.

When considering children in the 5 – 7 years age range, the ability to read and write cannot be assumed. One way this is often addressed in user interfaces is through the use of icons or symbols. Naranjo-Bock, a UX researcher at Yahoo, has found that basic media icons like 'Play', 'Pause' and 'Stop' are generally understood by children due to their familiarity with media players [6]. Naranjo-Bock also found that simple symbols like arrows and pencils are normally understood. She observed that children at age 5 become more interested in reading words matched with an icon, but notes that if possible, the labels should be reduced to one single descriptive word.

Font choice is an important consideration for text to be read by children who have recently learnt to read. Strizver [7], a typography consultant, recommends a font as similar as possible to those used when teaching children to read. As adults, it is easy to recognize the several different fonts shown in Figure 5 all to be the character "a". However, this skill is not present in those new to reading. Strizer also recommends using lower case, large, sans serif fonts in colours that contrast against the background.

Figure 5: Lower case "a" in a variety of fonts.

Gelman [8] notes that children enjoy visual and auditory feedback when interacting with a user interface. In contrast, adult users may get annoyed by this and prefer feedback only at the point of success or when they have done something wrong. An element that Gelman has found successful for both children and adults is the idea of including an 'Easter egg', some unexpected treat, within the user interface experience.

The attention span of younger children is likely to be considerably shorter than for adults. Therefore, if the user has to wait for a response, or if it appears the system has stopped working, they may lose concentration.

Fine motor skills will be mostly developed by the time a child reaches 5 years [9]. However, some children may have difficulties manipulating small or awkward shaped objects. This will restrict the minimum size of objects that require manipulating.

Colour plays an important role in designing for younger children. Colour vision is one of the last senses to develop. Bright colours are generally more appealing to younger children, since they stand out more than dimmer ones. Although by age 5 the vision senses will be fully developed, children of this age still generally prefer bolder colours [6].

2.2 Review of previous studies

This section reviews four studies exploring the potential advantages of tangible interfaces over screen based equivalents.

Xie et al [10] performed an exploratory study comparing three types of interface (traditional, graphical and tangible) while completing jigsaw puzzles. Pre-study questionnaires were completed, then pairs of participants used one of the interfaces to complete jigsaw puzzles, and then answered another questionnaire afterwards. Observations were taken on the time spent completing the puzzles and the total time playing. This study involved 132 children and found there was no significant difference between the enjoyment levels reported by the children using the three interfaces. A significant number of children replayed the puzzle using the tangible and physical interfaces.

Horn et al [11] conducted a study at the Boston Museum of Science using an interactive computer programming and robotics exhibit. The exhibit had a tangible and a graphic user interface and 260 museum visitors were observed, and 13 family groups interviewed. Researchers set up either a tangible or graphical interface to program a robot with simple instructions, then observed museum visitors. Their results showed that visitors found both interfaces equally easy to use. However, visitors were more drawn to the tangible interface and it was observed that the tangible interface led to more collaboration between group members. There was no significant difference in the time spent on either interface.

Sapounidi and Demetriadis [12] compared the use of the T_ProRob tangible programming language and a graphical equivalent to program an NXT Lego robot. The graphical equivalent was programmed via a mouse. They performed this study with 61 children in three age groups (5-6, 7-8 and 11-12) within a school setting. Observations were taken during the interactions and questionnaires and interviews were used to record perceptions. The children were shown the two interfaces and asked which they would rather use. Their answers were recorded. Then the children were presented with the instructions for programming the robot using both interfaces, and afterwards asked which they enjoyed most. The next stage was to get the children to complete two tasks using one of the interfaces, and then complete two similarly difficult tasks with the other interface. Then the children were asked again which interface they preferred. Their results showed that the tangible language was more attractive, since the majority of children wanted to use it after first sight. The younger two age groups (5-6 and 7-8) reported finding the tangible interface easier to use, but the opposite was found for the 11-12 year old group. It was also observed that the tangible interface allowed for a more group style of programming, with multiple hands being able to manipulate the blocks at one time.

In a further study, Sapounidis et al. [13] performed similar testing, but also video recorded the sessions to record number of mistakes and time taken to complete a task. Five groups were used: 6-7, 7-8, 9-10, 10-11 and 11-12 years. Sapounidis et al. found that, apart from the oldest group, tasks were completed more quickly with the tangible interface. Notably, the younger the age group the larger the difference between the two interfaces. The tangible interface also outperformed the graphical interface regarding the number of errors in programs and success of debugging.

2.3 Review of teaching children to program and tangible programing languages

The idea of creating technology to teach children to program is not new. Logo is an educational programming language created in 1967 [14]. Logo is most remembered for the turtle, which started as a physical robot attached to a computer and then was converted into an on-screen graphic. This allowed the drawing of geometric shapes.

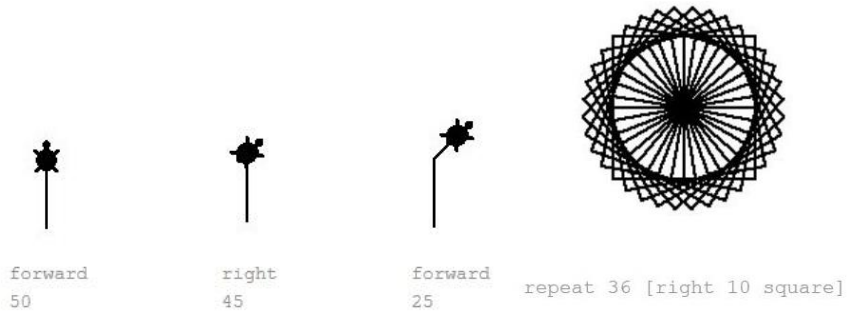

Figure 6: Basic logo commands to control turtle [15]

It became clear to those teaching Logo that children under 10 were struggling to learn to program. They believed that one of the barriers was the user interface. For this reason in the mid-1970s, Perlman created two alternatives to inputting instructions via the keyboard [16]. The first was the 'button box': this was a set of 4 individual boxes that could be connected, each box having a different set of buttons to control the turtle. More boxes could be added together as the child mastered more complex controls.

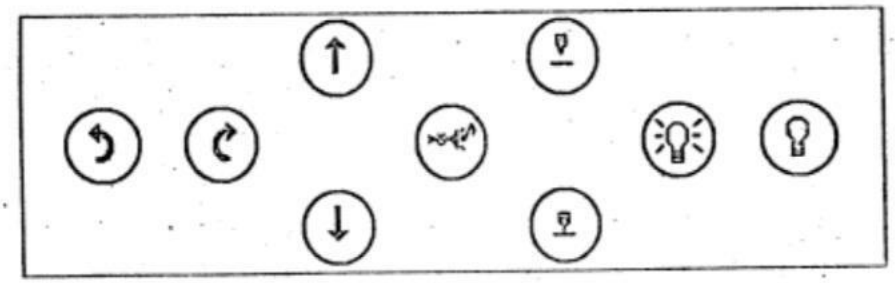

Figure 7: Perlman's "Action Box"; one of 4 button boxes [16]

The second was "The Slot Machine" which had long rectangular boxes with slots into which the child could place instruction cards. On the boxes was a button that, when pressed, ran the instructions on the cards in the order they were arranged.

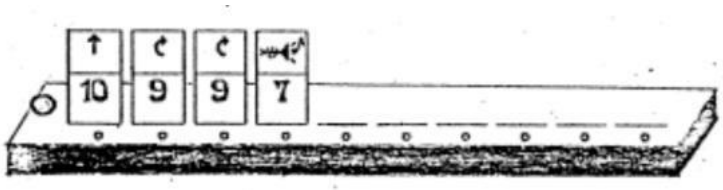

Figure 8: Perlman's "Slot Machine" [16]

Interestingly, both of these tangible user interfaces make use of metaphors to indicate the meaning of the tile or button being used, e.g. an upwards arrow indicating the turtle would move forwards.

Reflecting on more recent developments, a number of tangible and non-tangible languages have been built in an attempt to make teaching children to program more effective. Tile based programming languages have become increasingly popular, for example, Scratch [17]. Using Scratch, the user can manipulate sprites by dragging tile-like objects around a screen. It is a very popular language with almost fourteen million registered users [18]. Scratch is designed for 8 to 16 year olds [19] and the majority of users are over ten. It is likely that children may use Scratch or a similar tile-based language, e.g. Blockly [20], at some stage while learning to program.

Lightbot [21] is another popular non-tangible programming language. The user moves a small robot around levels lighting up blue tiles. Lightbot can be played as an app on a tablet or in a browser window. Interestingly, Lightbot has very few instructions and on the first level the user only has access to two (forward and light a tile up). More instructions are then added as levels progress, which is similar to how Perlman incrementally added more buttons to her button box [16]. Lightbot also limits the maximum number of instructions used in a sequence; in the early levels this is limited to twelve. The concept of incrementally adding to the instructions while limiting the length of the sequence of instructions is an interesting one.

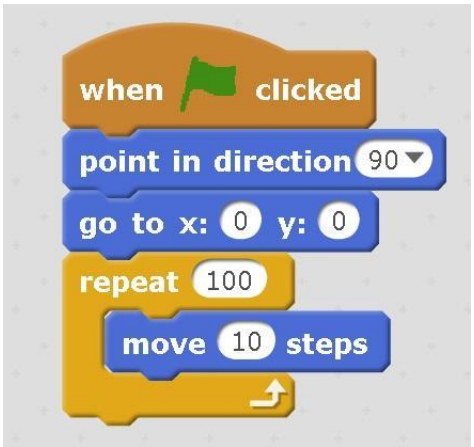

Figure 9: Scratch programming instructions [17]

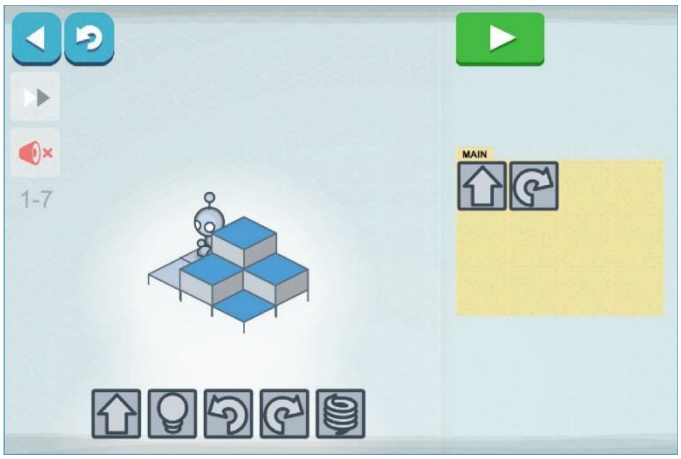

Figure 10: The Lightbot game interface [21]

Current TPLs can be broadly categorised by 2 characteristics:

1) The method the system uses to identify the blocks: one family uses computer vision to identify tiles and their order, the other family uses in-built electronics allowing the blocks to uniquely identify themselves.
2) What the TPLs control: this can be split into one set controlling robots, and the other set being games running on tablets.

One of the most popular commercially available tangible languages is Osmo coding [22] (which started as Strawbies [23]) which is similar to Lightbot since it involves controlling a character (a monster) around a grid-like level to specified squares (collecting strawberries). The method of user input is different: Osmo coding uses tangible blocks to give input to the app, unlike Lightbot. Osmo is a game system that uses tangible play with an iPad.

Figure 11: Strawbies [23] on the left and Osmo [22] coding on the right

The original Strawbies game made use of the TopCodes computer vision library. This library, created at the Tufts University [24], recognises ninety-nine unique small circular codes and can return an ID number and the location of each code. Strawbies is not the only tangible programming language to make use of the TopCodes library; TPLs Tern [25] and Quetzal [26] also use it. Both Strawbies and Tern also use jigsaw-like puzzle pieces, ensuring the user connects the tangible blocks in the manner intended.

Figure 12: A TopCode code

Figure 13: Tern's [25] and Strawbies [23] jigsaw like puzzle pieces

Cubetto [27], a TPL aimed particularly at younger children, has the users place blocks into fixed slots in a wooden board to control a little wooden robot on a grid-like mat. The Cubetto mats stand out by their colouring and graphics (Figure 14).

Figure 14: Cubetto TPL and mats [26]

One of the most recent developments in the tangible programming world is the Project Bloks [28], a Google research project building a platform to allow others to create tangible programming experiences for children. This will potentially allow those with less technical experience to be able to easily create tangible interfaces.

3. Professional considerations

3.1 Project compliance with BCS Code of Conduct

The interim report addressed project compliance with the BCS code of conduct and outlined potential ethical issues. This section considers how the ethical issues identified in the interim report were addressed.

3.1.1 Public interest

Point 1a “Have due regard for public health, privacy, security and wellbeing of others and the environment” – The design and implementation of all physical elements of the project was done in a manner that would not endanger the health of children. All data collected during the evaluation phase was anonymised and stored in a password protected file to protect the privacy of those involved.

Point 1b “Have due regard to the legitimate rights of Third Parties” – The app makes use of a number of third party libraries, credit is given in Appendix H and in the code base. No images in this report or the app infringe any third parties’ copyright.

Point 1c concerning discrimination. During the evaluation phase all those who volunteered to take part were accepted so no discrimination took place in the selection process.

3.1.2 Professional Competence and Integrity

Point 2c concerning continued professional development. In completing this project a number of new skills were gained, and while learning these new skills the most up to date standards were used where applicable.

Point 2d concerning complying with legislation. The data gathered during the user testing and evaluation phases complies with the Data Protection Act 1998 [29].

Point 2e: “Respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work.” – Feedback was sought from the project supervisors and peers. This feedback was reflected on and acted on appropriately.

3.2 Ethical considerations and ethical compliance

One of the secondary objectives of the project was to run small scale study with children aged 5-7. While this study did not take place due to time limitations, an in-depth examination of the relevant ethical issues was conducted, which resulted in a draft ethics application and discussion of the relevant issues (Appendix I).

As an alternative, a small scale study was run with adults. This meant that an ethical compliance form was required instead, shown in Appendix B.

4. Requirements analysis

A list of requirements for the project was created before the design stage. The requirements were split into the three main components (tangible objects, tablet computer and robot).

4.1 Tangible block requirements

Functional requirements

1.1 Blocks must be created for the following instructions:

- Move robot forward 30cm.
- Turn robot ninety degrees right on the spot.
- Turn robot ninety degrees left on the spot.
- An iterator.
- Make the robot produce a noise.

1.2 Each variation of block must be uniquely identifiable.

1.3 Blocks must be able to be connected in a sequence.

1.5 In a sequence of blocks, blocks must only be able to be connected in a syntactically correct manner.

1.4 The iterator block must be able to hold one or more of the other instruction blocks. These other instructions will be repeated the number of times defined on that iterator block.

1.6 There must be a clear block/button that can be used to run the sequence of instructions.

Non-functional requirements

1.7 Blocks must be durable enough to survive a drop from 1.5 meters.

1.8 Blocks must be big enough that the intended user group can easily hold and interact with them.

1.9 Blocks must be small enough that a sequence of at least ten can be executed at once.

1.10 Block use should be differentiated by colour.

1.11 The colouring of the blocks should appeal to children aged 5 – 7 years old.

1.12 Block meaning should be conveyed by both symbols and words.

1.13 Blocks should easily be switched in and out of a sequence.

1.14 The target age group should be able to easily understand the behaviour of an instruction from the appearance of the block.

1.15 Blocks must be built such that if they are broken there are no sharp edges or live electronics exposed to the user.

1.16 When the run button is pressed, either audio, tactile or visual feedback should be given.

Extended requirements

1.16 Create an instruction block that is the equivalent of an if statement, that checks if the robot's path is blocked and performs a given instruction if true.

4.2 App requirements

Functional requirements

2.1 Be able set up a connection with the robot component.

2.2 If no connection is made, notify user and repeat process or exit.

2.3 Have two modes (tangible mode and screen mode) that will be selectable.

2.4 Once in either mode, allow the user to exit and select a different mode.

2.5 In tangible mode - when the run sequence button is pressed - capture an image of the tangible bricks in the camera view and convert these to robot equivalent instructions.

2.6 In screen mode, allow the user to select instruction block equivalent tiles on the screen and create sequences by dragging and dropping instructions.

2.7 In screen mode, display a run sequence button equivalent to the tangible version.

2.8 In screen mode - when the run sequence button is pressed - convert the selected instructions into robot equivalent instructions.

2.9 Once instructions are in robot equivalent, transfer these to the robot wirelessly.

2.10 If the instructions sent to the robot would cause it to leave the current task layout, only send instructions to the robot that are within the layout.

2.11 Accept acknowledgment of instructions from the robot and re-send instructions if no acknowledgment is received.

2.12 The app must be able to store a digital form of the tasks that the robot completes.

2.13 The app must be able to be set to a specific task.

2.14 On running a sequence of instructions, the app must calculate if the instructions complete the current task successfully.

2.15 The following data will be recorded:

- Time on task.
- The number of times code is run to complete a task.
- If the task was completed.
- Interface used.
- Instructions used.

2.16 The app must allow new tasks to be added.

2.17 It must be possible to view data recorded by the system during task attempts.

Non-functional requirements

2.18 Button presses in the app must be responsive (i.e. some action must be shown within half a second).

2.19 On screen, blocks are the same colour and shape as the tangible equivalent.

2.20 Colour schemes used in the app design must be suitable for children aged 5 – 7.

2.21 All code will be written in a structured manner with all substantial functions being documented.

2.22 Backups will be made of the code base on at least a weekly basis to prevent loss of work.

2.23 The app must be built in a way that future instructions can be easily added.

2.24 As close as feasibly possible the screen interface will imitate the tangible interface

4.3 Robot requirements

Functional requirements

3.1 Will connect to a tablet wirelessly.

3.2 Listen for incoming instructions via wireless communication.

3.3 On receiving an instruction, perform appropriate action.

3.4 On performing an action, send an acknowledgment to the app wirelessly.

3.5 For the following instructions the following actions should be performed:

- forward – robot moves 30cm forward in a straight line.
- right – robot turns 90 degrees right on the spot.
- left – robot turns 90 degrees left on the spot.
- beep – robot makes an audible noise.

Non-functional requirements

3.6 The robot will be battery powered while running.

3.7 Must be able to run for at least 2 hours on battery alone.

3.8 No electronics or sharp edges will be exposed on the body of the robot.

3.9 The front of the robot will be easily identifiable.

3.10 Operation will be possible within 10 metres of the control system.

3.11 The appearance of the robot should be designed with target age in mind.

3.12 The robot will run reliably, i.e. work as intended with the app in normal circumstances.

5. System Design and Implementation

5.1 Choice of technology for system components

Technology decisions formed the first phase of the system design.

For the tangible blocks, an image recognition method was considered against building electronics into each block. It was decided to use an image recognition method for the following reasons:

- Cost - In using an image recognition system, no electronic components are required for each block thus saving costs.
- Block design – As the blocks won't contain electronics, the block design is less restricted.
- Easier use – No electronic connections are required between blocks.
- Durability – Without electrical components the blocks will be more durable, since when dropped there is no chance of electrical components becoming disconnected.
- No need to charge blocks – As the blocks have no electrical components there is no need charge or replace batteries in the blocks.

Once image recognition had been selected as the method for the tangible block recognition, the type of image recognition had to be selected. TopCodes [24] was selected as it was designed for tangible object recognition, has successfully been used in multiple projects, and provides the required libraries for Android and iOS operating systems.

iOS and Android were both considered as operating systems: Android was selected for the following reasons:

- Ease of access to suitable hardware – Access to a tablet running Android was easier and less costly than iOS. This also holds for the future where, if multiple tablets were required, sourcing Android tablets is possible at a lower cost than iPads.
- Quicker and cheaper app publishing process – If required, publishing in the Android store is both cheaper and quicker than the iOS equivalent.
- Previous experience – Some previous experience was held in developing in the Android environment which reduced development time.

For the robot, the Arduino prototyping board was selected for the following reasons:

- Quality and quantity of tutorials – A large number of high quality tutorials are available for the Arduino. This is of significant importance as little experience was held in this area.
- Large number of libraries – A large number of libraries are available for a wide variety of areas, making development quicker and more efficient since not all work must be completed from scratch.
- High number of compatible hardware components – Meaning there is a high chance that components can be combined to sufficiently meet the requirements.
- Large community – There is a large active community interested in the Arduino; for this reason it is possible to get help from forums, for example, from very knowledgeable people.

The wireless communication standard chosen to communicate between the tablet and robot was Bluetooth for the following reasons:

- Well supported – Android has built in APIs [30] for Bluetooth use and Arduino has suitable hardware to enable simple Bluetooth communication.
- Sufficient range – Bluetooth provides sufficient range [31] for the tasks that the robot will need to perform.

Figure 12 illustrates this combination of technologies, providing a high level view of technology interaction, and the sequence of those interactions.

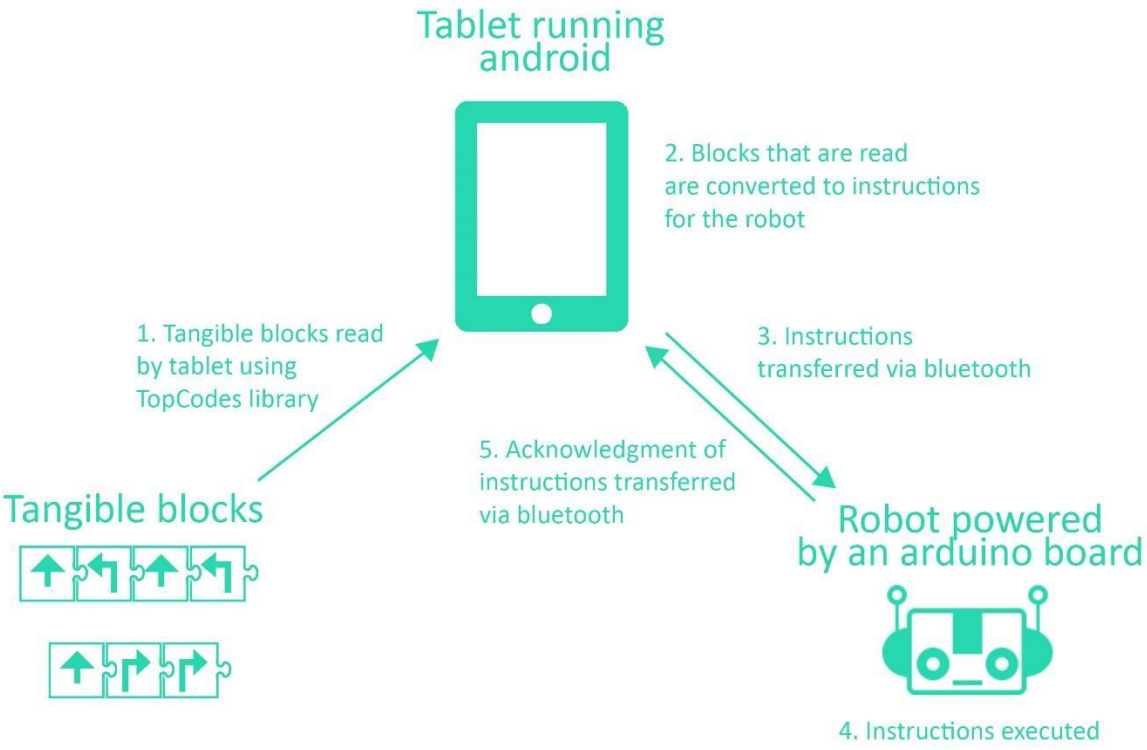

Figure 15: Demonstrates the technology selected and high level interaction between components.

5.2 App design

5.2.1 App architecture

A three tier architecture was selected to design and implement the app. A data tier was used store and to control access and updates on data held relating to tasks and task attempts. A logic tier captured the logic of the application. A presentation tier controlled the display and captured user input. A three tier architecture was selected as it provided a logical separation between tiers, allowing (if required) a tier to be independently changed. For example, if the database engine required to be changed this would only affect the data tier. The simplicity of a three tier architecture suited the size and the complexity of the app.

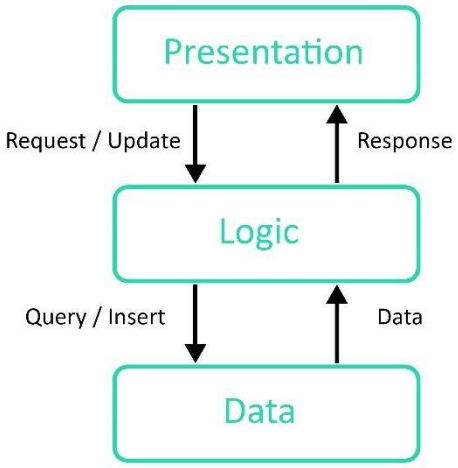

Figure 16: Overview of three tier architecture

5.2.2 Use case diagram

Having decided on a three tier architecture, the next step in the app design process was to create a use case diagram. The use case diagram provided a quick and high level way to transfer from the requirements to a more concrete description of the roles those using the app would take, as well as a high level description of the app

functionality.

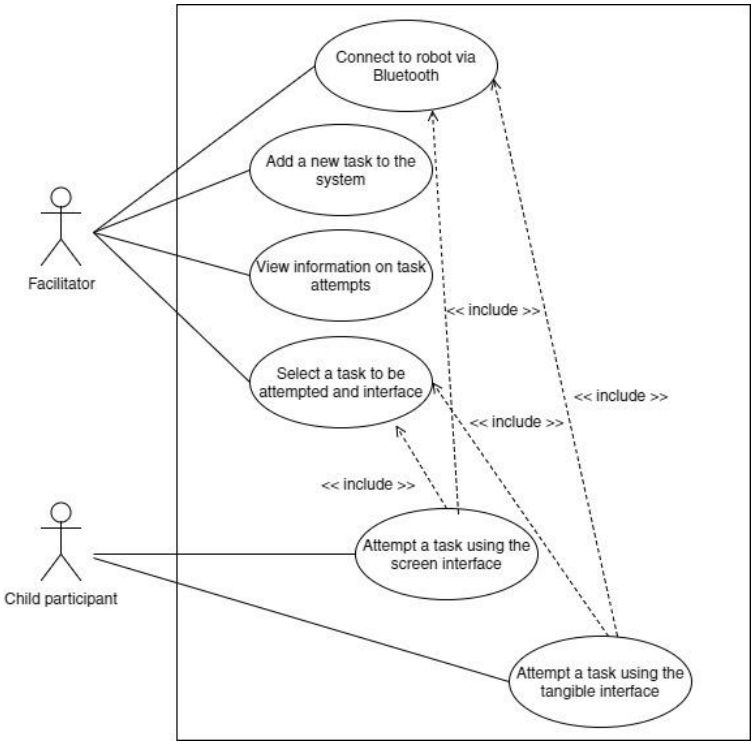

Figure 17: Use case diagram for the app

5.2.3 Use case scenarios

Using the use case diagram (Figure 17) and the app requirements, a set of use case scenarios were created in order to explore each of the use cases in more detail. One use case scenario is shown below; the rest can be found in Appendix C.

Use case: 1. Connect to robot via Bluetooth
Primary actors: S: System F: Facilitator
Preconditions: Robot should be turned on and app started
Basic flow of events: 1. S: As the app starts the system attempts to connect with the robot via Bluetooth. 2. S: Connection is made, the system stores this connection to be used later.
Alternative flows 2a. S: The connection can't be made as the tablet does not have Bluetooth capabilities. The user is informed that this app can't be used with this tablet. 2b.1 S: Bluetooth is not enabled on the device. The user is given the option to turn it on. 2b.2 F: Selects to turn on Bluetooth. 2b.3 S: Re-attempts to make connection. 2c.1 S: The connection can't be made as the robot can't be found. The system prompts the user to ensure the robot is turned on and has battery charge. 2c.2 F: Turns on robot. 2c.3 S: Re-attempts to make connection.

5.2.4 Activity diagrams

Activity diagrams were created to capture some of the decisions and workflows that became apparent while building the use case scenarios. They also provided a good reference which was used while constructing the app to

understand the logic of the system. A single activity diagram was created for the use cases “Attempt a task using screen interface” and “Attempt a task using tangible interface”, since it was found that there was duplication between the two diagrams. One activity diagram can be seen below; the rest are in Appendix C.

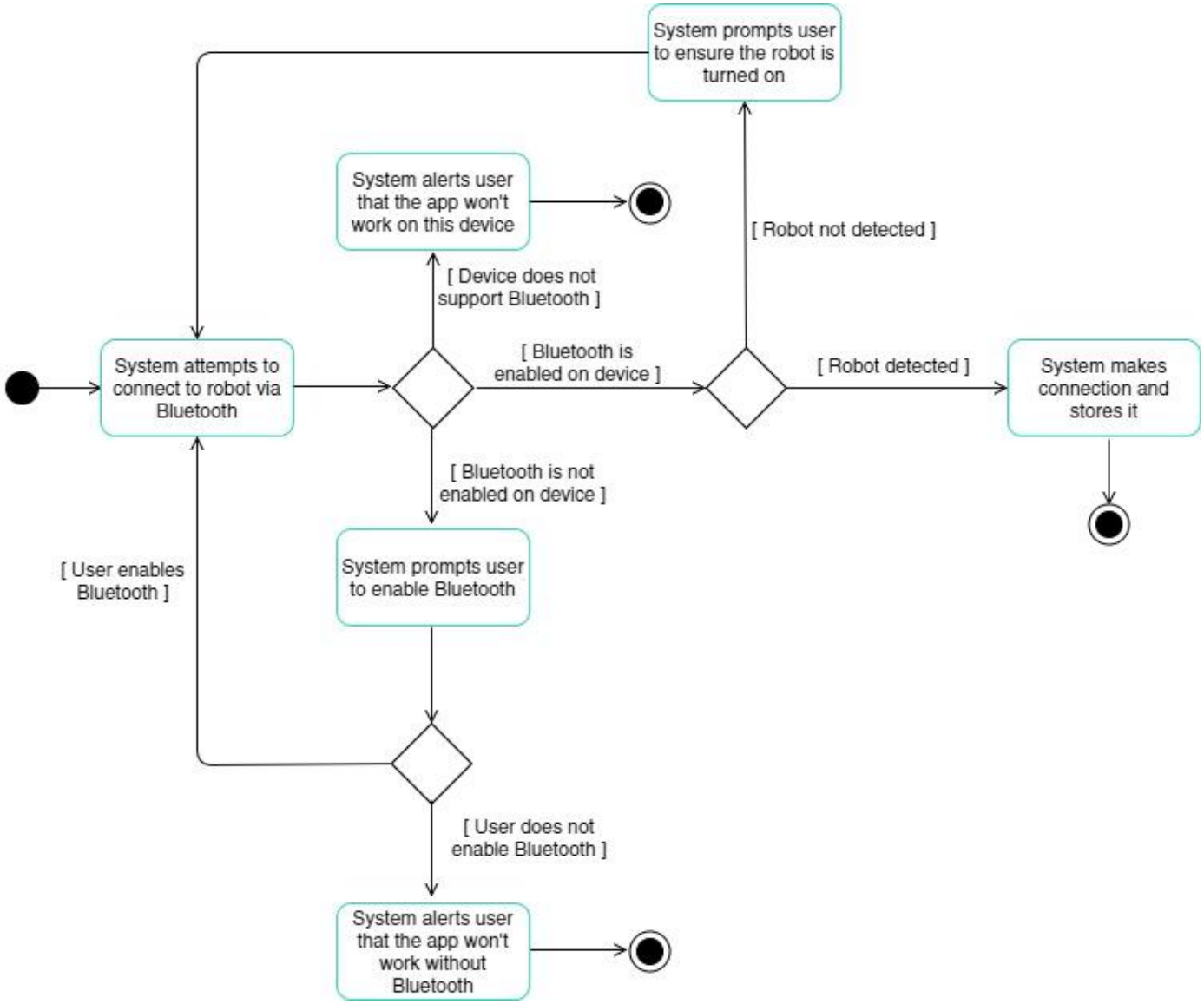

Figure 18: Activity diagram for use case “Connect to robot via Bluetooth”.

5.2.5 Logic tier

Having explored the app behaviour at a high level, design of the logic tier could begin. A high level class diagram showing the basic classes and their relationships was first created.

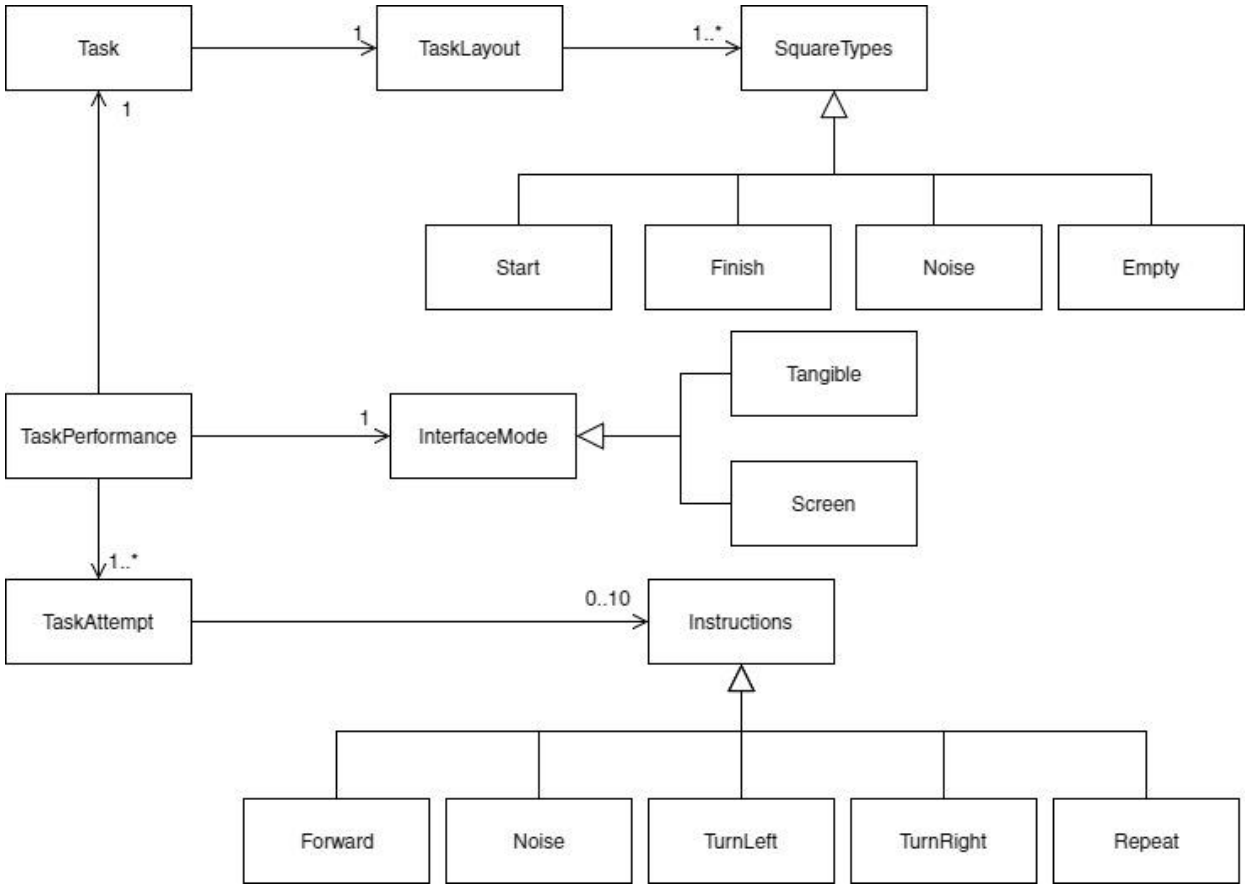

Figure 19: High level class diagram.

This high level class diagram was developed into a lower level class diagram. A number of the classes became enumerations as they didn't require the ability to store or manipulate data (e.g. the SquareTypes classes).

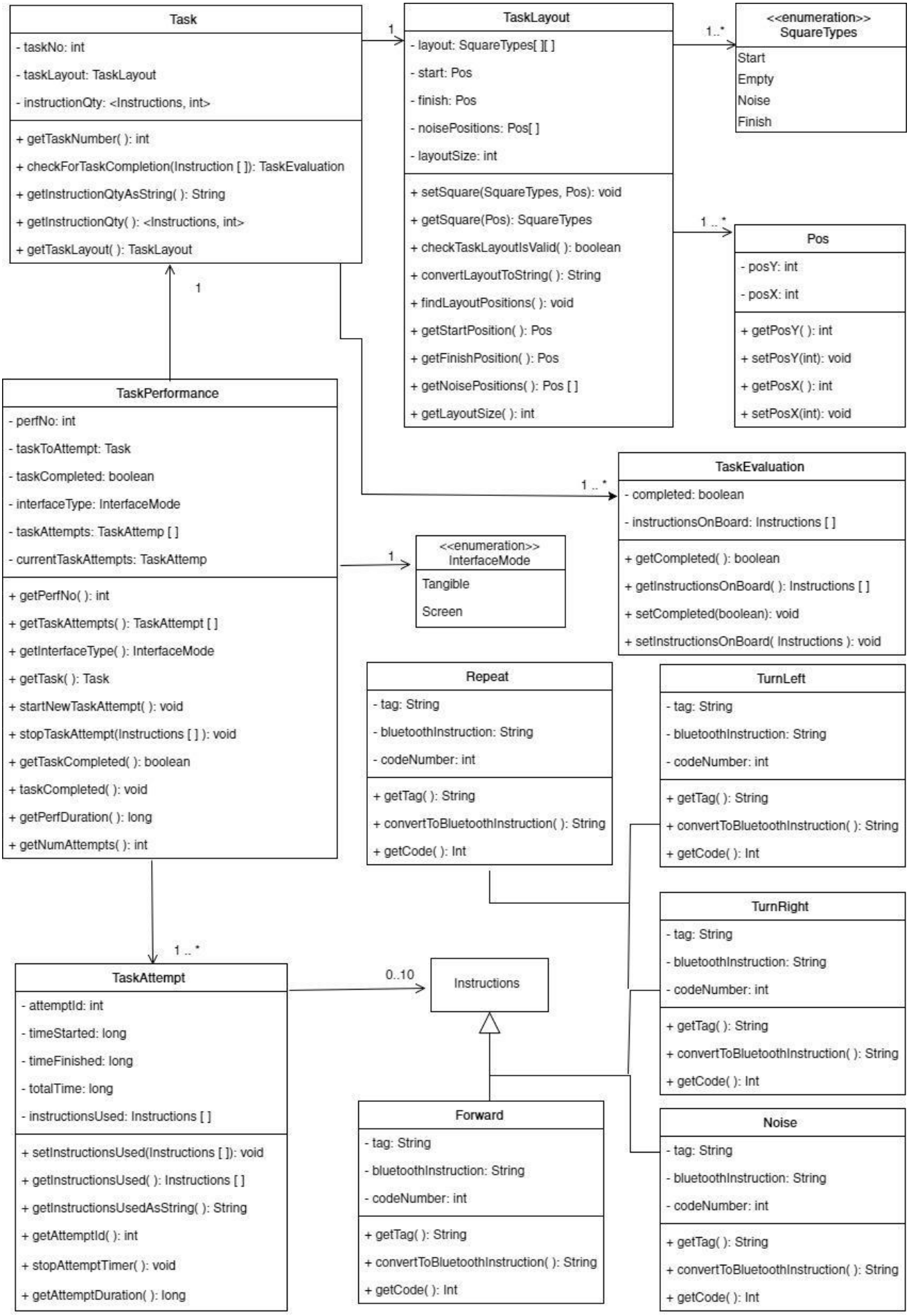

Figure 20: Low level class diagram.

The low level class diagram was used when implementing the code and the same structure is still very evident in the final code base.

5.2.6 Data layer

The data layer was required in order to store data regarding the tasks as well as each time the user tried to complete a task. This was called a "task performance" and each time the user sent instructions to the robot was called a "task attempt". The entity relationship diagram (ERD) (Figure 19) was created. This data is normalised to Boyce Codd 3rd normal form [32] to reduce data duplication and anomalies. The ERD was then used for reference when creating the database. SQLite [33] was selected for the database engine, because it comes with Android and provided more than sufficient functionality to implement the designed database.

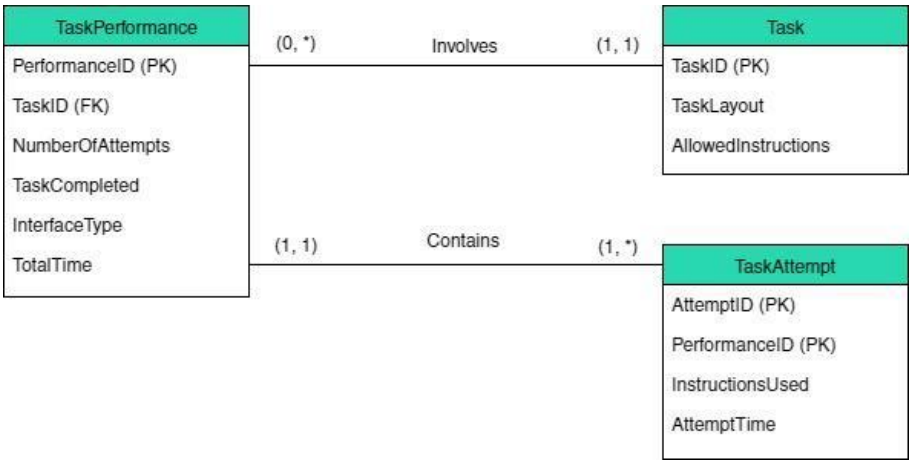

Figure 21: Entity relationship diagram for the system database.

In order to control the database, two main classes were designed. A simple class diagram can be seen in Figure 22.

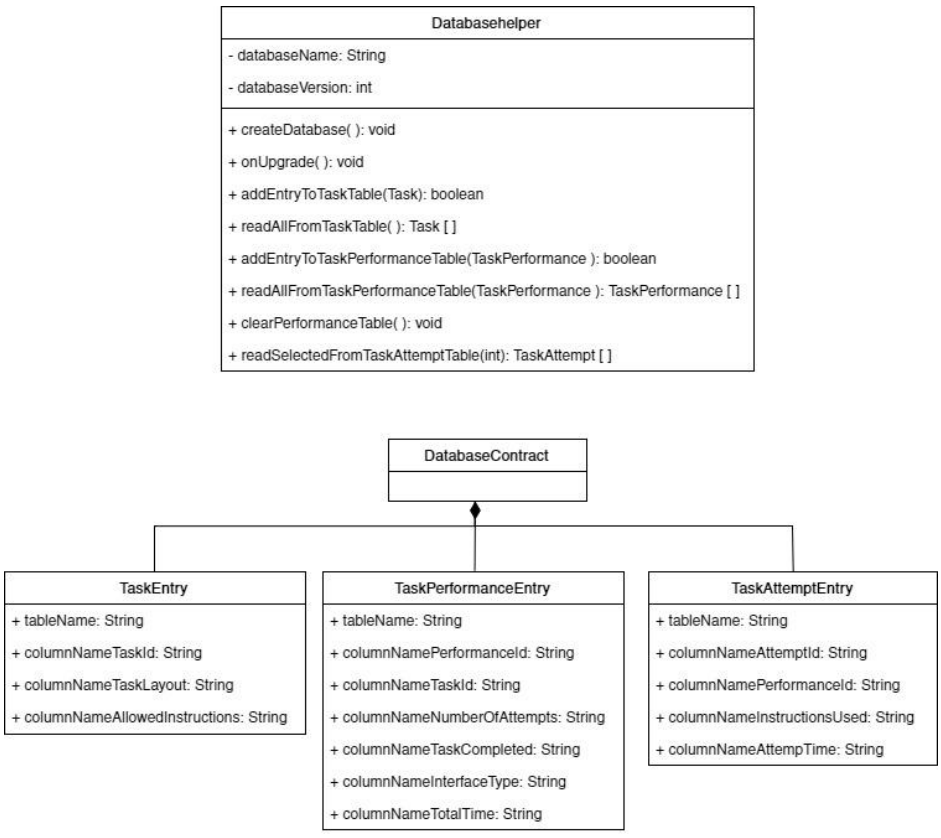

Figure 22: Class diagram for data tier.

The database contract class was used to store the table and column names for the database. The classes in a composition relationship were intended to be inner classes of the DatabaseContract. Should a table of column names require changing, this is the single location which needs updating. The database manager class was used to manage insertions, updates, deletes and queries to the database. Should the database engine be changed in future, this class would be the only one requiring updating. It acts as an interface between the database and the logic tier.

5.2.7 Presentation tier

To design the presentation tier, simple wireframes were created for each required screen. The requirements document, use case scenarios and activity diagrams were used to ensure that the wireframes could fulfil the app requirements. These wireframes were combined into a single diagram with annotations indicating the sequence in which they would appear during app use. Development of the presentation tier was partly concurrent with the design of the other tiers, robot and tangibles. As all parts of the system interacted, some overlap and iteration in the design process occurred.

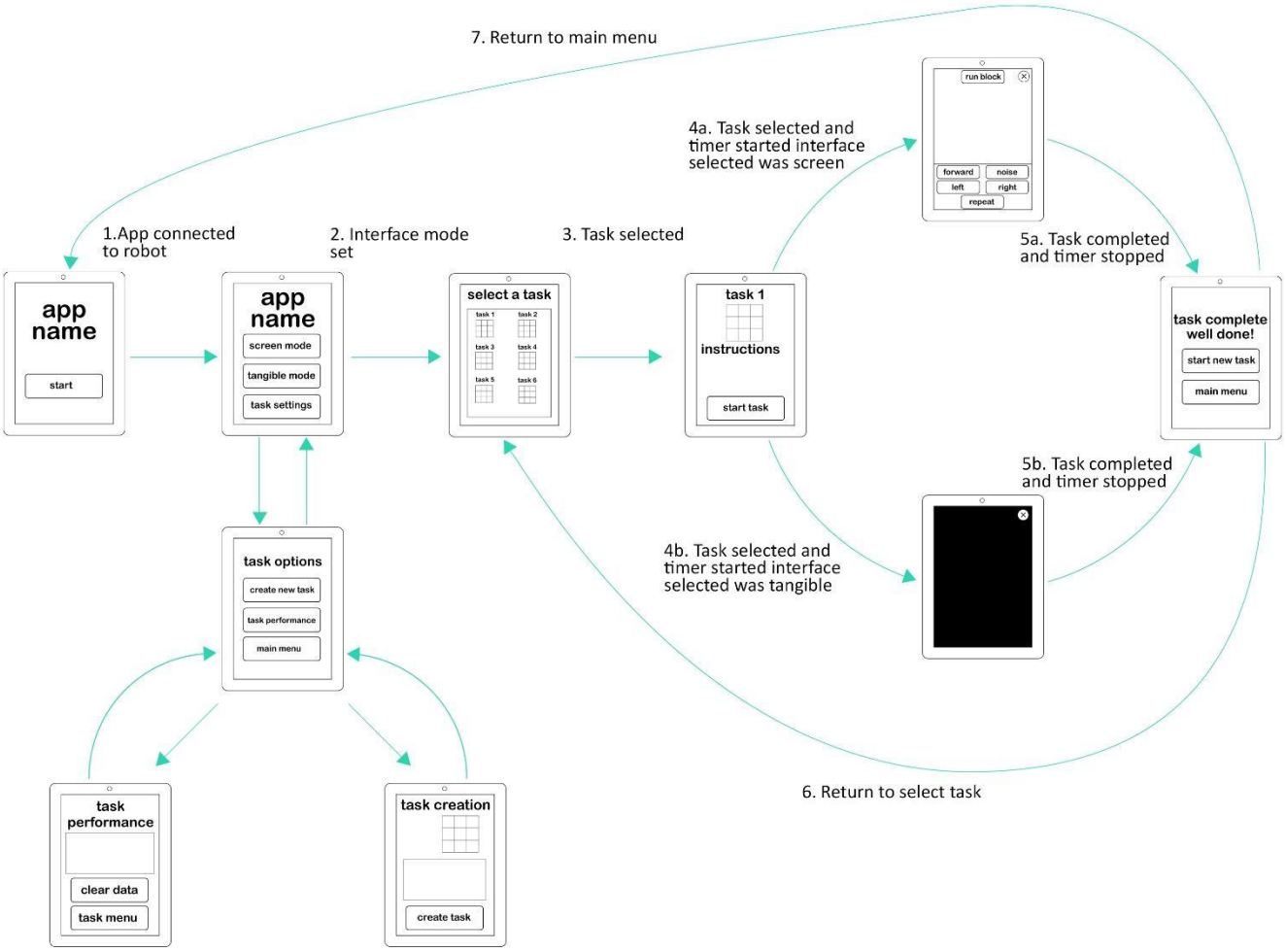

Figure 23: App wireframes and sequences

Using the wireframes for layout and knowledge gained from background literature, screen designs were then created. While only small sections of the app would be used by child participants, it was decided that the whole app would be themed for the participants rather than the facilitator. This is reflected in the colours used and the use of lower case lettering. The app was named “TICA”: Tangible Interface Comparison App. Donald Norman’s [34] famous design principles (visibility, feedback, constraints, mapping, consistency and affordances) were used.

Figure 24: Start Screen

Figure 25: Main Menu

A plain start screen is used; the system sets up the Bluetooth connection while this screen displays. Any issues forming the Bluetooth connection are shown on this screen. The main menu is designed for simplicity and makes use of icons on the buttons to help understand functionality.

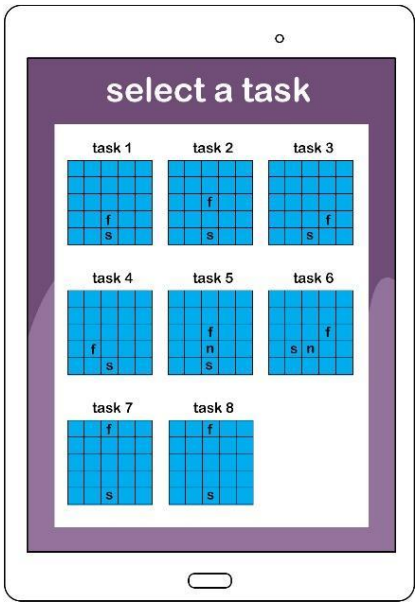

Figure 26: Task Selection Screen

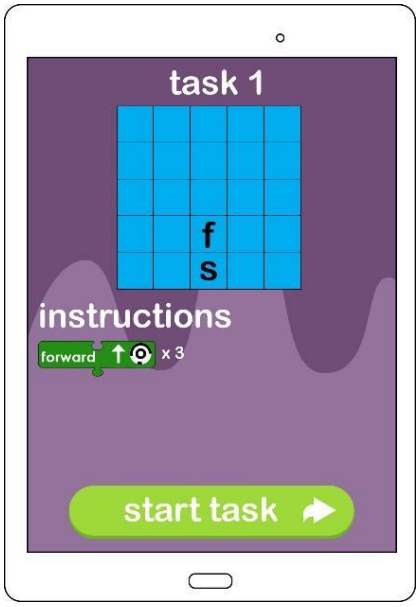

Figure 27: Task description screen

The task selection screen uses thumbnails of the task boards; this is done so the user does not have to remember each task number and can instead make use of recall, which is a principle of good design [35]. If there are more tasks than fit on the page, the area will be scrollable.

The task description screen gives the facilitator the opportunity to set up the task layout and present appropriate instructions.

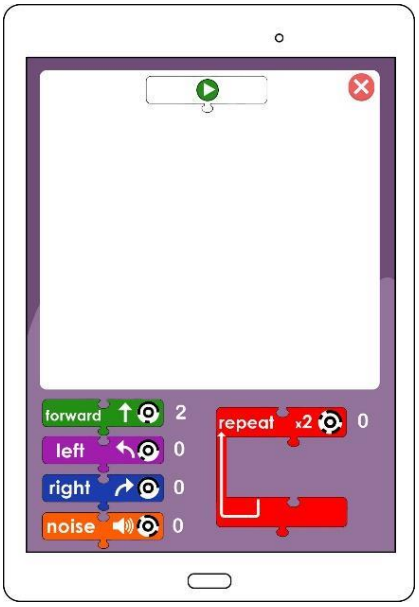

Figure 28: Task attempt screen interface

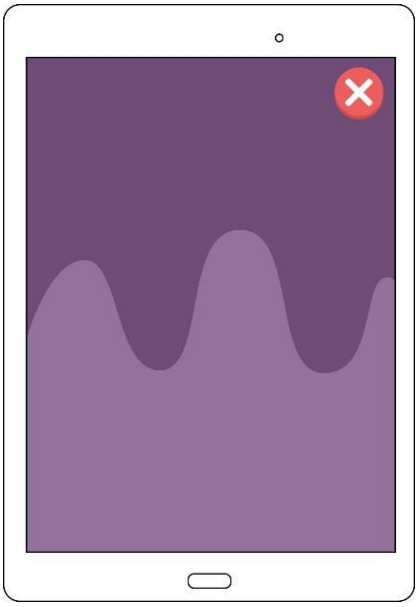

Figure 29: Task attempt tangible interface

The tangible task activity is created to be as close to the tangible set up as possible. The tangible screen is kept empty to avoid distractions. Both screens have an exit button; when pressed the system will check if the user really wants to stop before returning to main menu. On running a sequence of instructions, a dialog will appear to inform the user the robot is running. On completion of the robot running, if the sequence of instruction was incorrect the user will see a popup window alerting the user to this, or if correct they will move to the task complete screen.

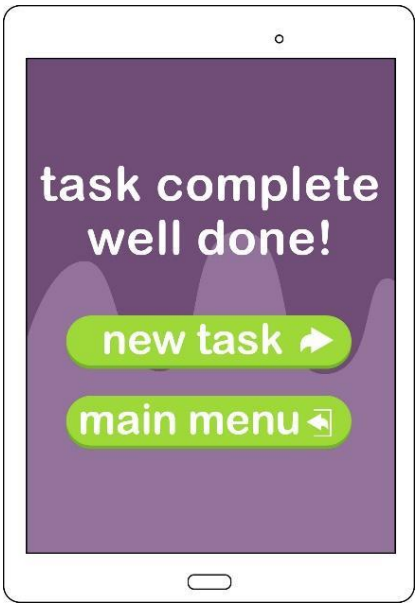

Figure 30: Task Complete Screen

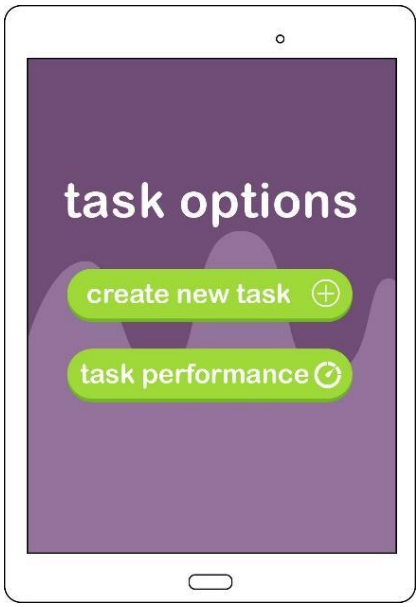

Figure 31: Task Settings Screen

The new task button on the task selection screen will return the user to the task selection screen. Menus are designed to be simple and clear.

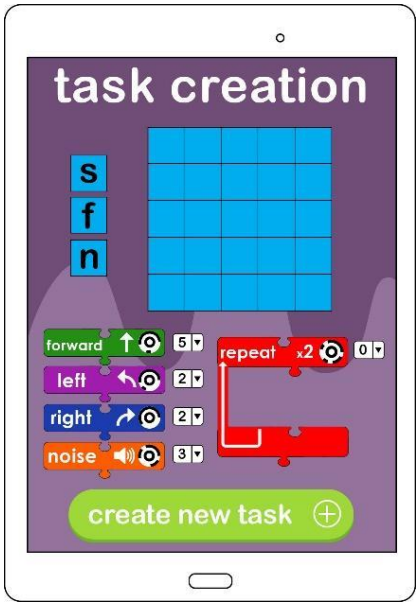

Figure 32: Task Creation Screen

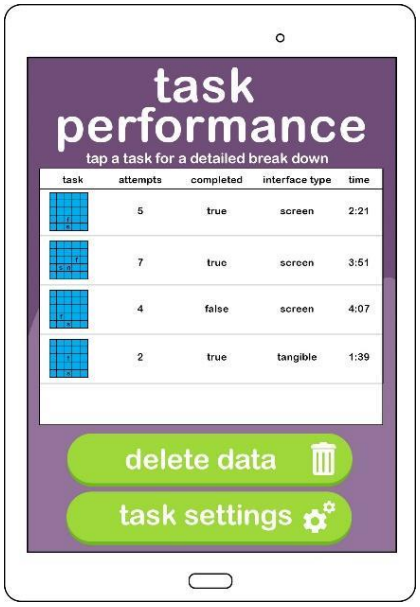

Figure 33: Task Performance Screen

Using the task creation screen, clicking on the "create new task" button will check that the task meets the requirements of a task, i.e. it has a single start and finish tile and at least one instruction allocated. Warnings will be given if this is not the case.

Tapping on a task in the task performance screen allows the user to view details on that task attempt (e.g. instructions used), shown in a popup window. The user can also delete all current performance data by tapping the "delete data" button. A message checks that the user is sure that they want to delete all data.

5.3 Tangible design

5.3.1 Initial tangible prototypes

A set of initial block prototypes were designed using key concepts from other TPLs (Section 2). These designs were modelled to help visualisation.

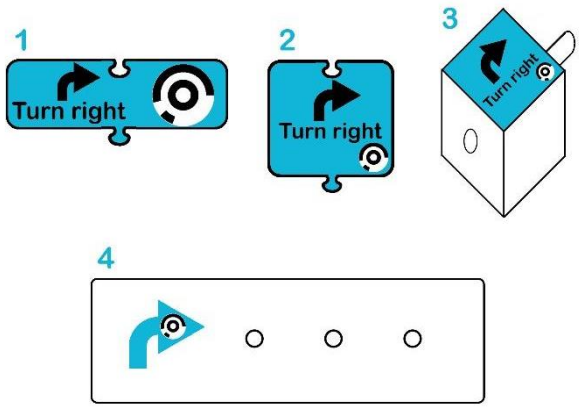

Figure 34: Initial Block Designs

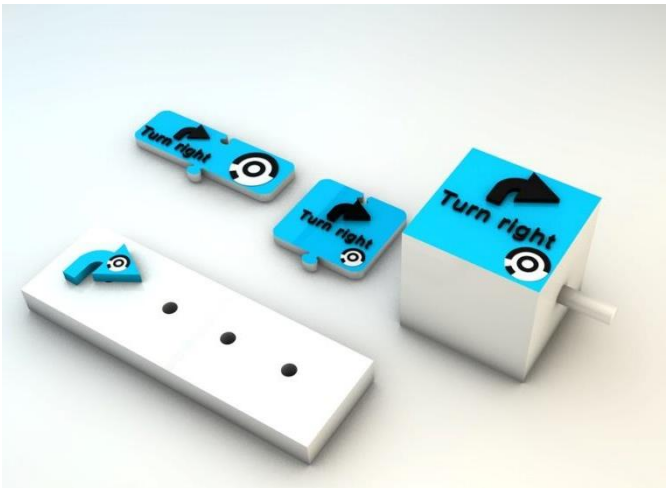

Figure 35: 3D models of initial block designs.

From the initial designs, physical prototypes were created using paper, cardboard and wood.

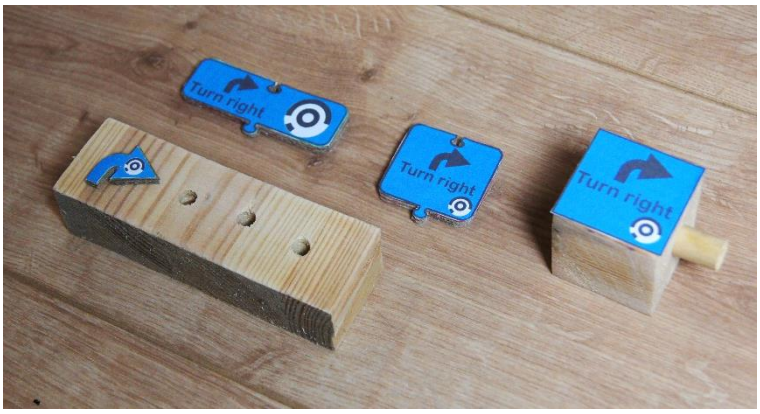

Figure 36: Prototypes of initial concepts

A number of positives and negatives for each block type were discovered. Comparing each, it was decided to base the designs on block number 1 (Figure 34). The main advantages are:

- A large number of instructions can fit into a small area.
- Shape allows a larger TopCode code, making identification more accurate.
- Lends itself easily to a repeat construct.
- High similarity to blocks used in Scratch [17] and Blockly [20].
- Easy to hold and move.

This was used as the basis for the first set of designs (Figure 37).

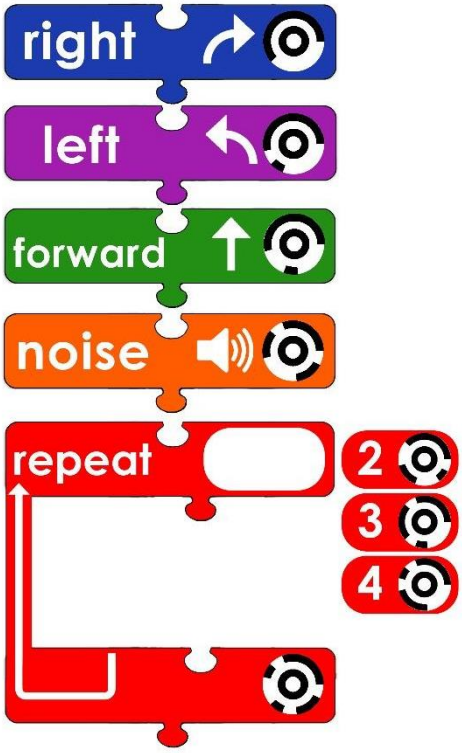

Figure 37: First draft tangible designs

The following design decisions were made based on background reading:

- Make the blocks wider, allowing both text and the symbol to be increased in size.
- Use single words for the right and left commands.
- Use all lower case lettering.
- Use symbols alongside words.
- Use colour to identify instruction type.

It became clear that the use of a repeat instruction with changeable digits indicating number of iterations would be fiddly. To overcome this, fixed iteration blocks were made. On reflection it was concluded that the removal of this variable was in fact an improvement, since the iteration concept may be challenging to the target age group.

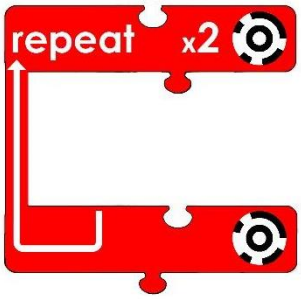

Figure 38: Redesigned iteration instruction

It was decided to use a 3D printer to build the tangible instructions. This provided a durable and safe material. 3D models of the instructions were created to be used for printing.

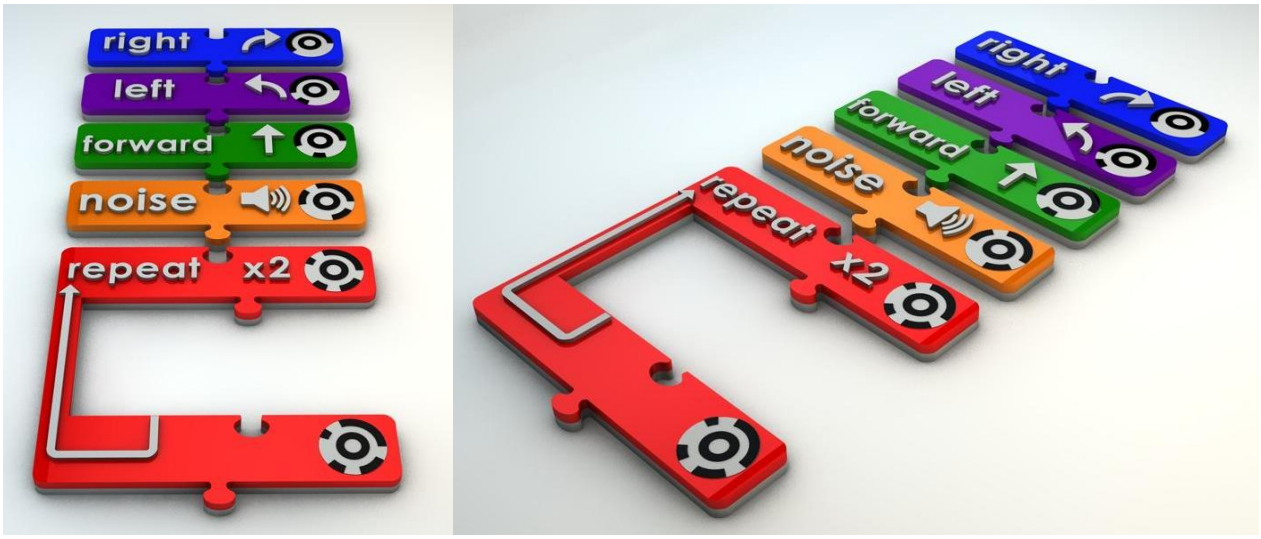

Figure 39: Render of 3D models of instructions

At a later point in the project, “if” and “if-else” instructions were designed in a similar style to those already created.

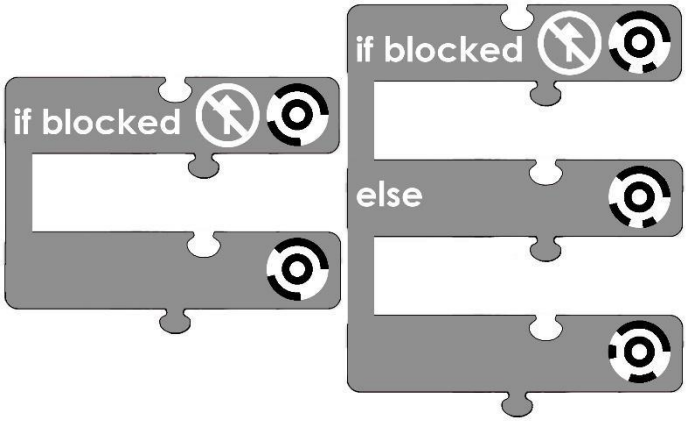

Figure 40: If and if-else instruction design

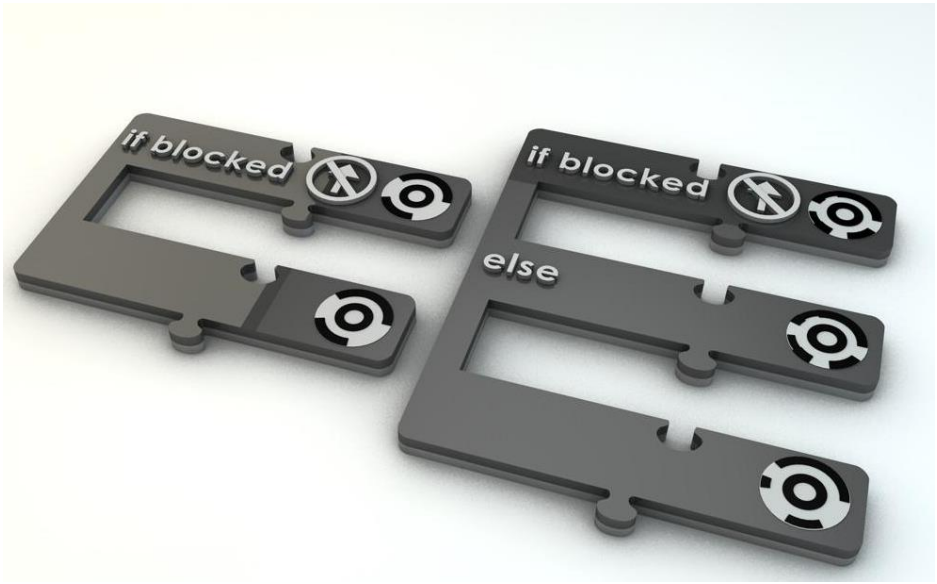

Figure 41: If and if-else instruction 3D render

5.3.2 Tablet holder and mirror design

A system for taking pictures of the tangibles was required so the system could identify them. In order to keep the tangible and screen interfaces as similar as possible, it was decided to implement a system similar to the Osmo [22]. This means the user does not have to pick up the tablet to take the photo. This system had two components: a mirror used to reflect the tangibles into the camera, and a holder for the tablet which incorporated a button that could be pressed to run the tangible instructions.

In order to add functionality to the play button, a remote Bluetooth photo taker was used. To the Android system the photo taker is seen as a Bluetooth keyboard, and when the button is pressed the Android system reacts as if the up-arrow was pressed. This interaction causes an image to be captured. The clicker was quite large, so it was decided to place this at the back of the case and attach a smaller button to be pressed.

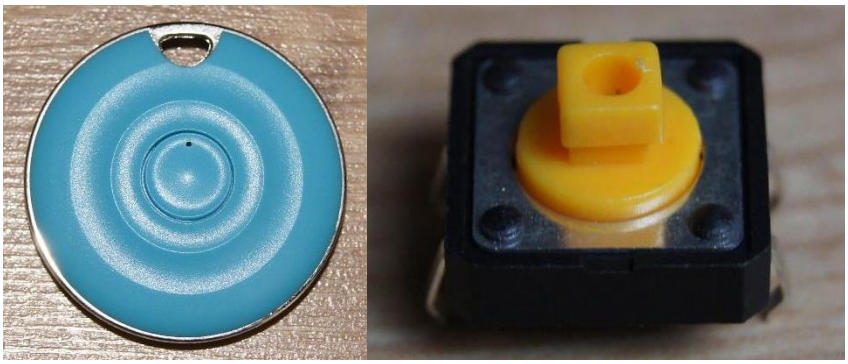

Figure 42: Photo taker and smaller button to be attached.

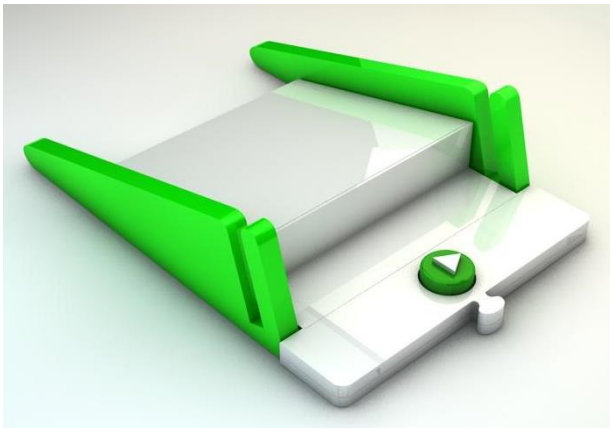

Figure 43: Play button and stand design.

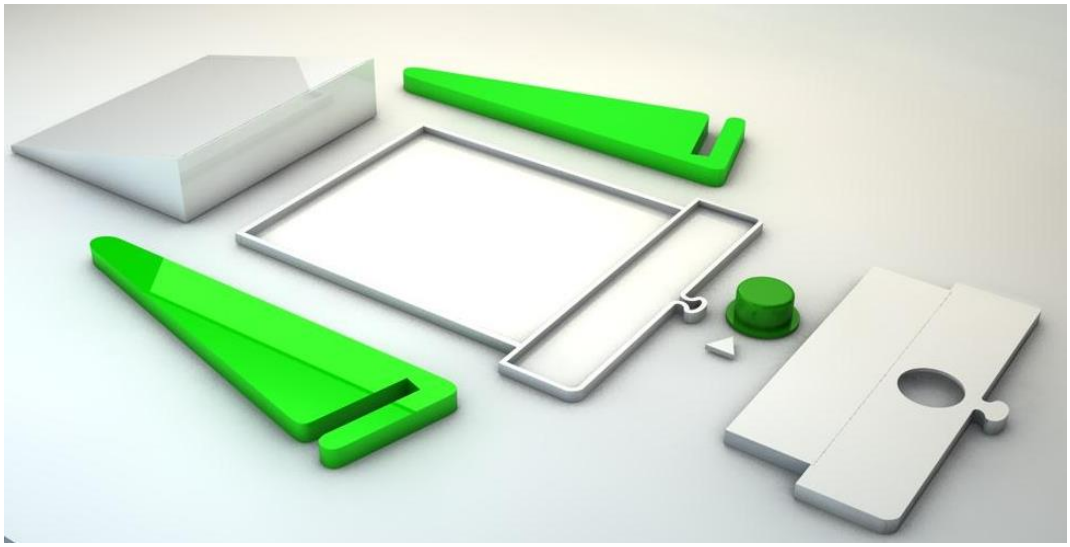

Figure 44: Play button and stand design in separate components.

A holder was designed that would contain a mirror to reflect the tangibles into the camera.

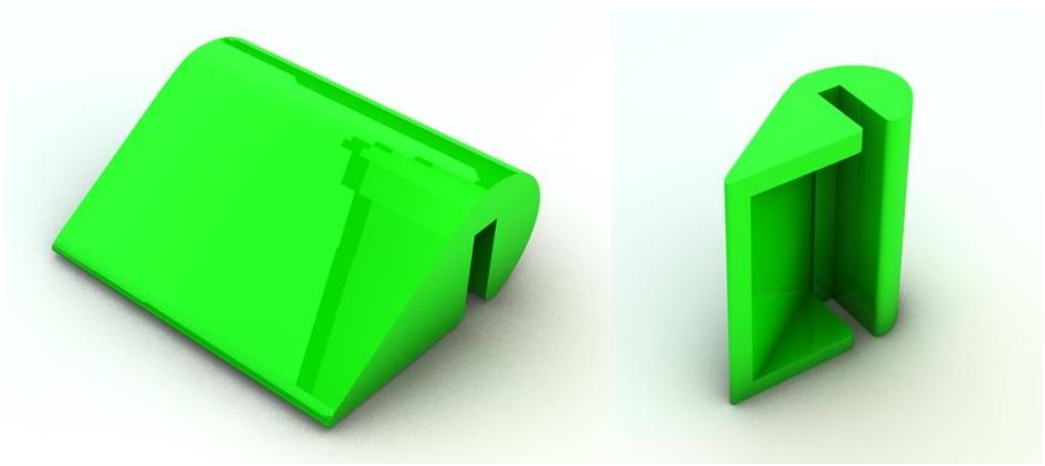

Figure 45: Design for mirror holder.

5.4 Robot design

5.4.1 Initial robot design

The initial robot design was virtually modelled, allowing the position of the components to be easily re-arranged (Figures 46-48).

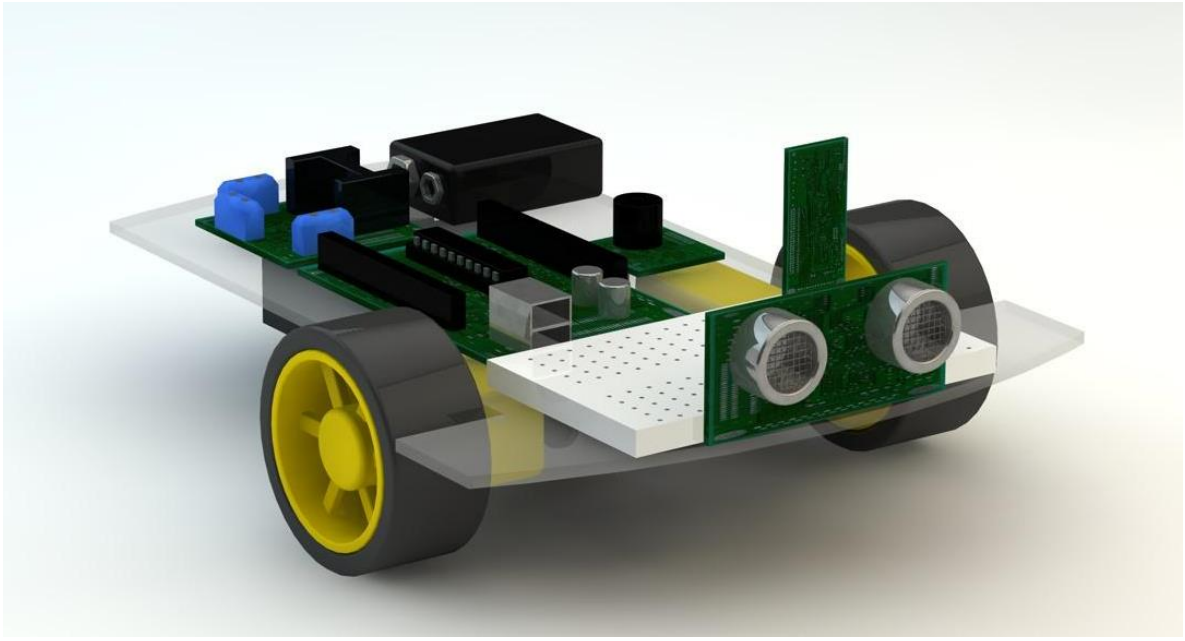

Figure 46: 3D model of initial robot design without a cover

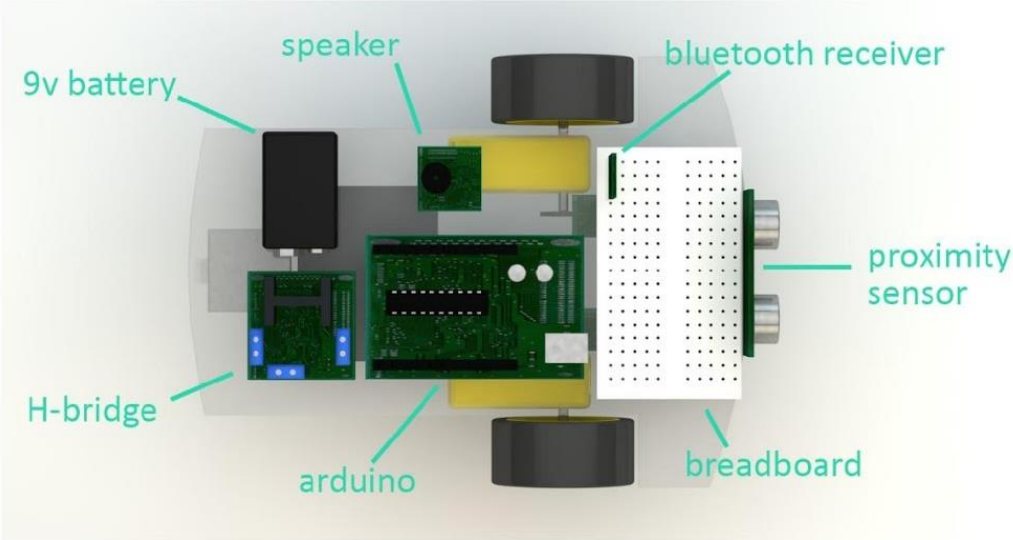

Figure 47: Design of the top of the robot with key components labelled

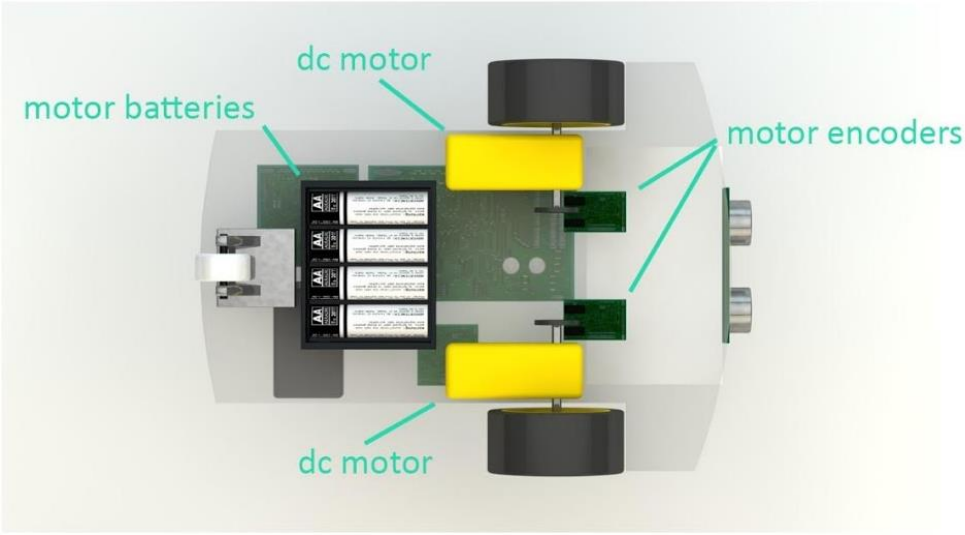

Figure 48: Design of the bottom of the robot with key components labelled

The robot case provides a cover for the electronics (requirement 3.8): initial design ideas are shown in Figure 49. The proximity sensors placed at the front resemble bug eyes, so a bug-like shape was used for the cover. The antenna were added to enhance this theme. The eyes, antenna and cover shape also indicate the front of the robot (requirement 3.9).

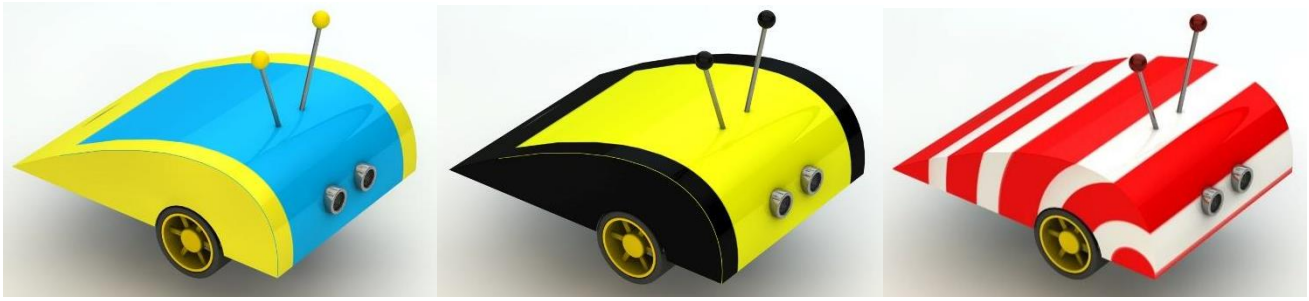

Figure 49: Three possible cover designs for the robot

5.4.2 Initial robot prototype

Having never worked with Arduino before, it was decided to build a prototype of the robot during the design phase. The hardware for the prototype was gathered and built to the design shown above.

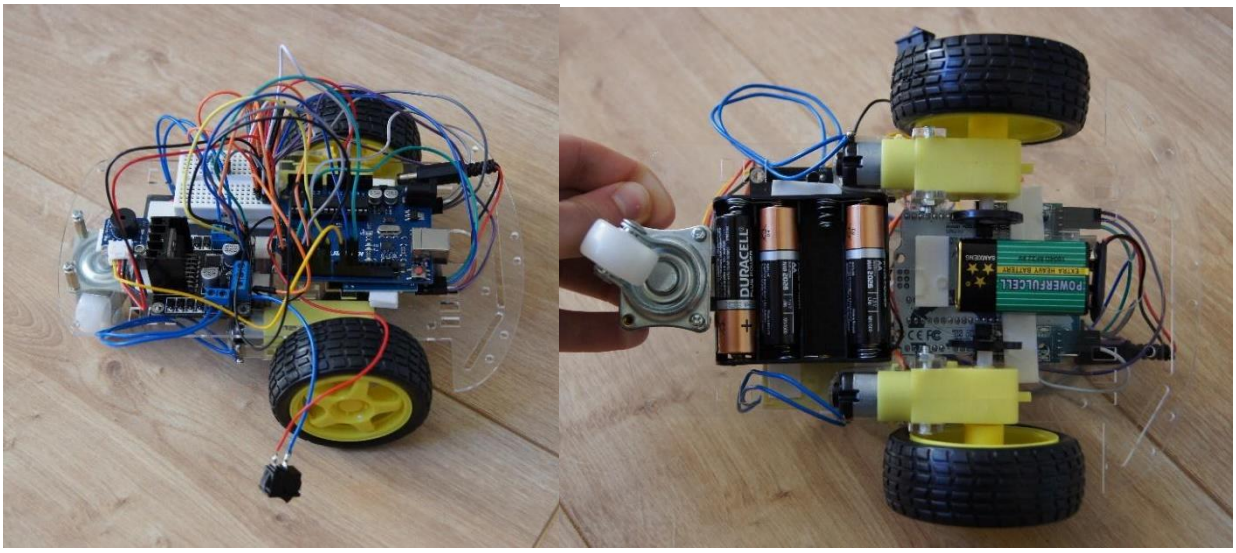

Figure 50: Robot prototype.

Arduino is programmed using C++, however it does not include a lot of the standard C++ libraries. All Arduino programs have two methods: setup and loop. Setup runs when the Arduino is first powered on and can be used to initialize program variables. Loop loops while the Arduino has power, and the body is executed. Programs running on Arduinos are often relatively short and are often written in a procedural manner. A procedural manner was used to design the code that would control the robot. A list of method definitions was created:

```
void setup()
void loop()
void forward()
void turn_left()
void turn_right()
void buzz()
```

The prototype robot design used DC-motors and motor encoders in an attempt to make the robot travel straight and move repeatable distances. Motor encoders use disks with slots in them and an infrared beam of light. Each time the beam of light is disturbed, a signal is sent to the Arduino.

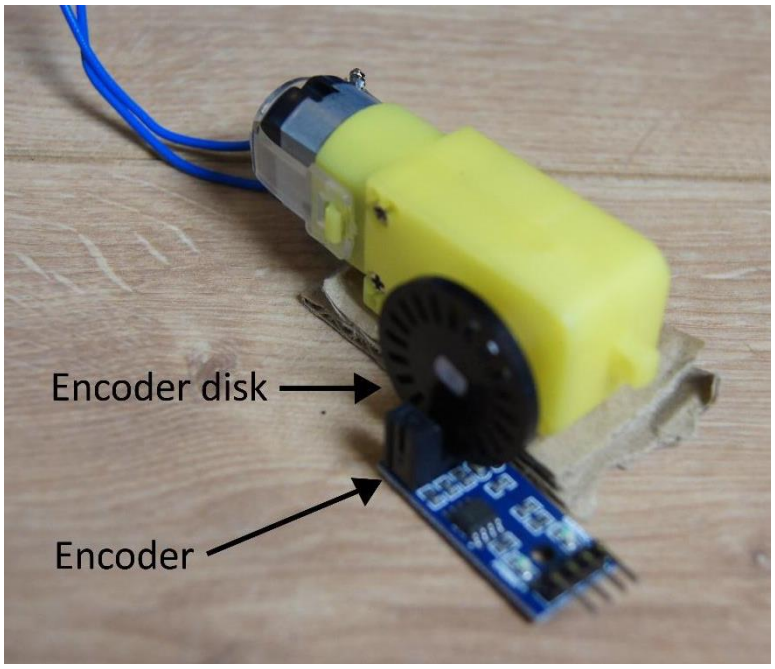

Figure 51: DC motor with encoder and encoder disk.

These signals can be counted to determine how far each wheel has travelled. The DC-motors are controlled with a pulse-width modulation (PWM) signal. A PWM signal is a square wave that can be used to control the speed of motors (Figure 52). When the wave is high the motor is on, and when low the motor is off. The Arduino can output PWM signals to control the motors.

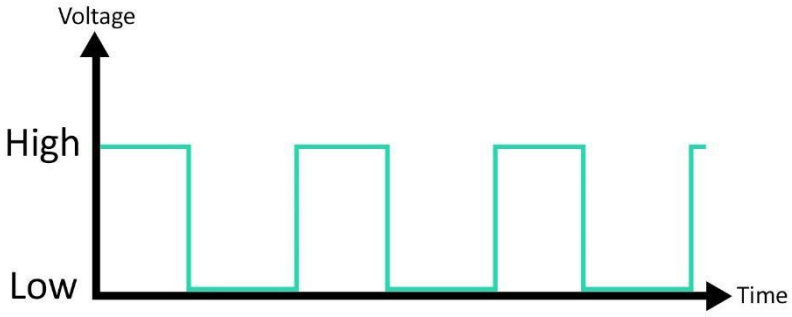

Figure 52: Example PWM signal.

To enable the robot to travel in a straight line, the left motor was set at a fixed PWM signal. Then the rotation of both motors was counted using the encoders for a set amount of time. After this time, if the right motor had travelled more than the left, then the power would be reduced to the left motor, or vice versa. However, in practice it is more complex. A common way of dealing with this is proportional integral derivative controller (PID controller) which is a feedback mechanism [36]. This was implemented but still the desired characteristics could not be achieved from the robot.

In order to improve the behaviour, the robot was re-designed to use stepper motors rather than DC-motors. Stepper motors are used when precise movement is required, for example in printers.

In conjunction with the stepper motors, a motor shield was also used. A motor shield contains multiple H-bridges and fits directly on top of the Arduino. Figures 53-55 show the re-designed robot.

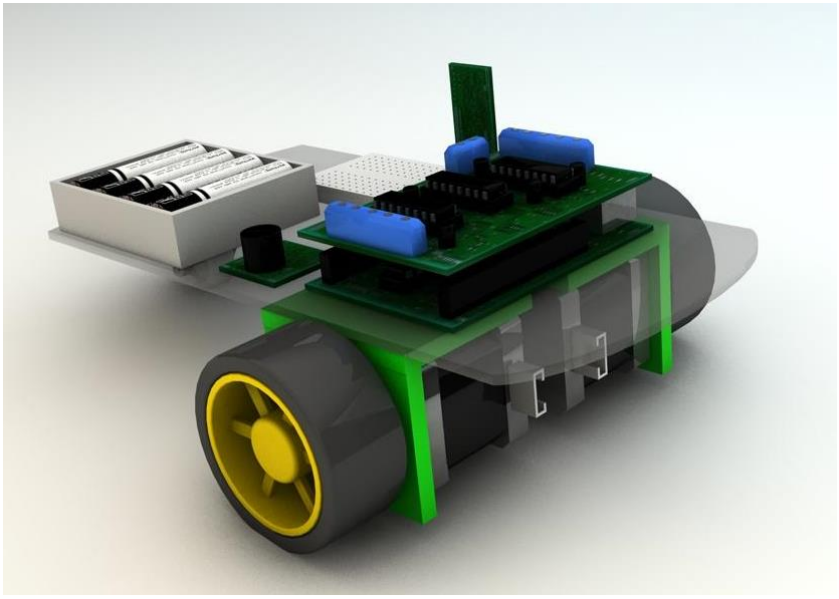

Figure 53: 3D model of final robot design without cover.

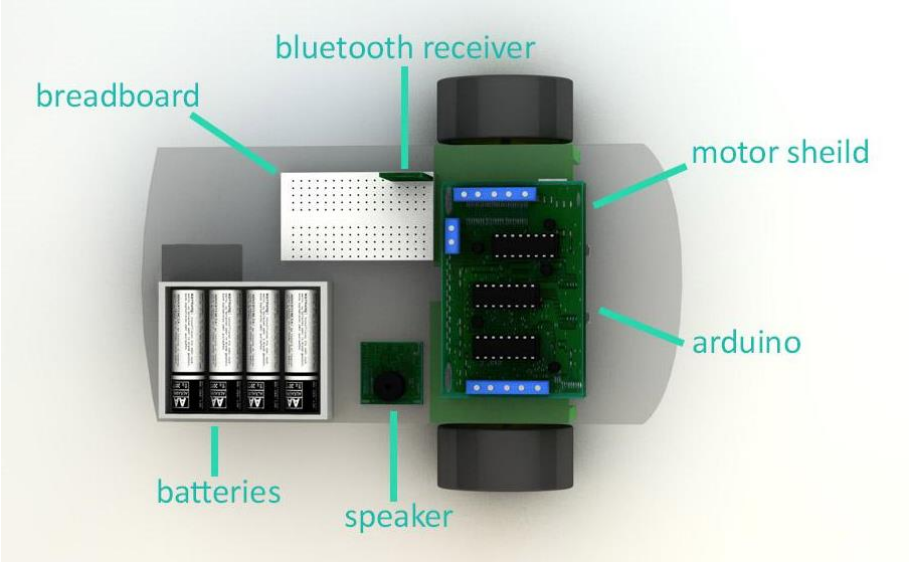

Figure 54: Robot from above with key components labelled.

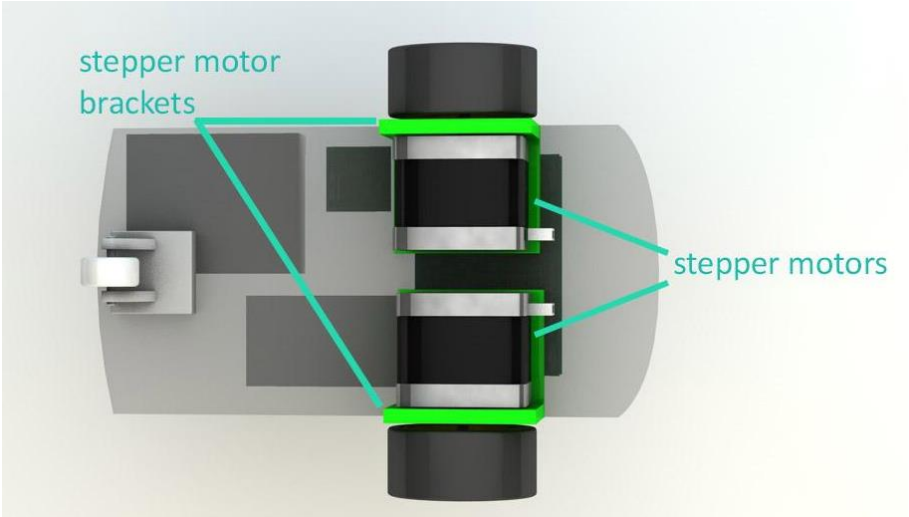

Figure 55: Robot from below with key components labelled.

The circuit of the robot design is also shown below.

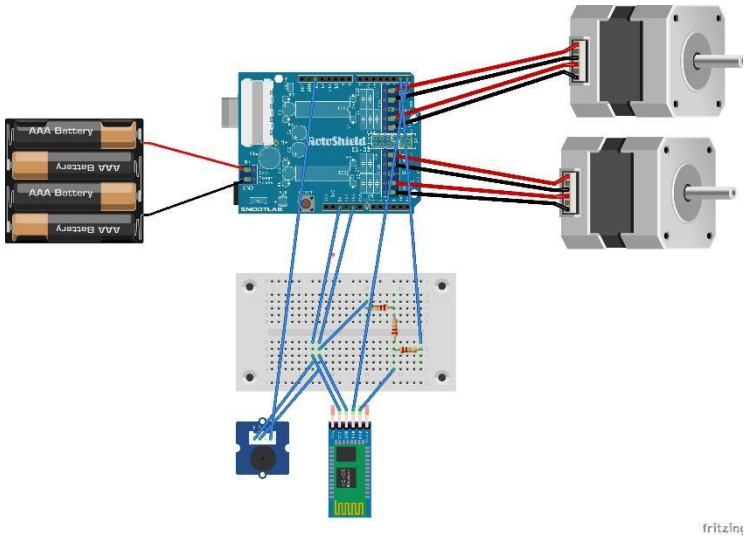

Figure 56: Robot circuit sketch.

The case was built from wood and 3D-printed plastic. The plastic had to be printed in two sections due to the limitations on the 3D printer capacity. The case materials were selected for the following reasons.

Wood	PLA plastic
Light weight	Light weight
Low cost	Medium cost
Relatively easy to work with	Can be curved
Safe for use with children	Available in multiple colours
	Safe for use with children

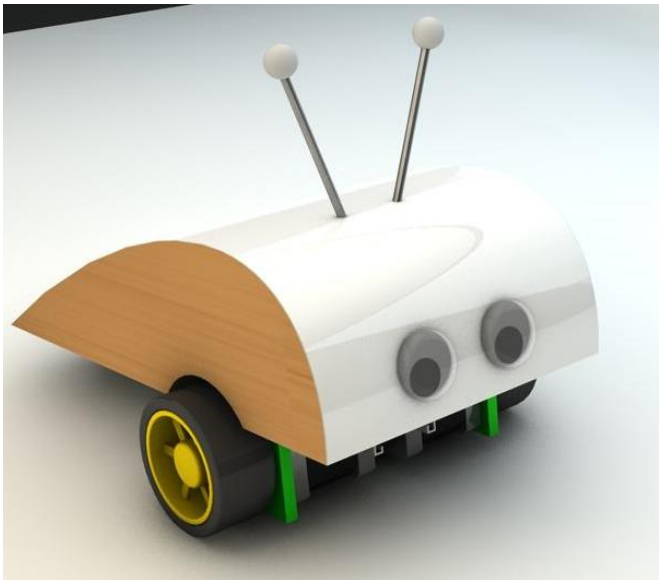

Figure 57: Final robot case design.

5.4.3 Tiles

Initially four tiles designs were required for the layout the robot would run on.

1. Empty tile.
2. Start tile.
3. Finish tile.
4. Noise tile.

The original design used letters to symbolise the different tile types.

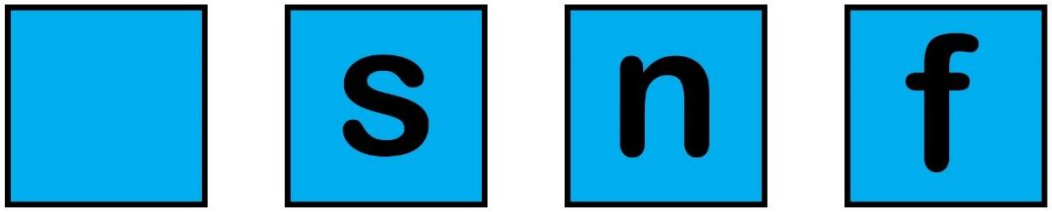

Figure 58: Original tile designs.

After reflection it was decided to use icons for the finish and noise tiles (Figure 59). Although the intended user

group may not recognise the finish flag, an icon will likely stay better in their memory than a letter.

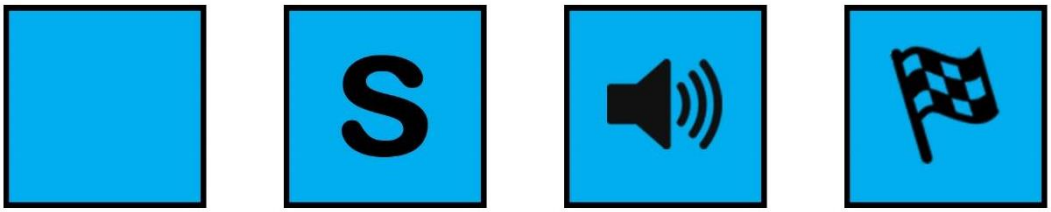

Figure 59: Final tile designs.

Later in the project a design was required for a tile that the robot could not enter.

Figure 60: Blocked tile design.

6. System implementation

6.1 App development

An overview of some critical elements considered during the app development stage is given.

6.1.1 Test driven development

Test Driven Development (TDD) was used throughout the development of the app where applicable. In TDD the following steps take place:

1. A unit test is written for each new feature that is to be added.
2. All the current unit tests are run, the new test should fail at this point.
3. Code is then written that should pass the test.
4. Run all the tests, all should pass but this checks the new code passes the test and hasn't changed the outcome of any of the other tests. If they all do not pass go back to step 3.
5. Refactor the code that has just been written to keep a high quality of code.

As a result of this, over 100 unit tests were written. These tests store the behaviour of the logic tier of the application. A real benefit of this is that in the future, after refactoring, running the unit tests will indicate if the behaviour of the program has changed as a result of the refactoring.

6.1.2 Version control

Version control was used to incrementally store the stages of development. This helped to identify the cause of bugs as earlier versions of the application could be checked until one without the bug was found, and then this would isolate at what point the bug was introduced. Git [37] was used for version control and GitHub [38] to host the repository remotely and privately. The use of GitHub allowed easy access to the code base from multiple machines and offered protection against hard drive failure that could occur if stored on a single computer.

6.1.3 Code style

The code style for this project followed those given for contributions to Android, which are mostly based on Java language rules [39]. Some key features include:

Write short methods – Methods should be short and focused, generally shorter than 40 lines.

Variable and method naming – All variables to be longer than a single letter when not used as a counter. Variable and method names chosen so their use is easily understood by the name.

Line length – Line length limited to 100 characters.

Define fields in standard places – Fields defined at top of classes.

Limit variable scope – Scope of variables limited to a minimum, thus helping readability.

Don't ignore exceptions – All exceptions to be caught and dealt with.

Use Javadoc comments – All non-trivial methods given a Javadoc comment.

6.1.4 Extending android classes

For the display portion of the system, classes were created extending Activities [40]. These were used for the main screens. Classes extending Fragments [41] were used when a pop-up was required.

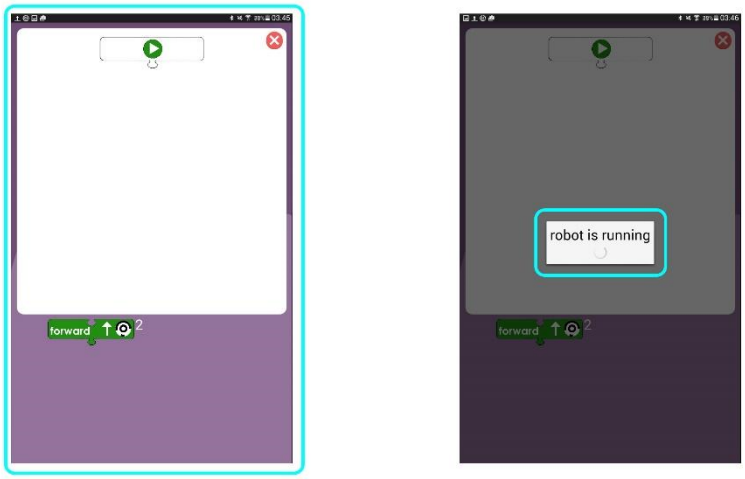

Figure 61: Activity highlighted on left and a fragment on the right.

A number of other Android classes were extended or interfaces implemented. For example, in order to get the functionality for dragging objects on the screen, DragListeners [42] were implemented and similarly ClickListeners [43].

Where possible, inner classes and anonymous inner classes were used to increase encapsulation, increase logical grouping and improve readability.

6.1.5 Multithreading

When working in Android the main thread is the UI thread and, unless otherwise programed, processes will run on this. However, running certain processes will cause threads to freeze. An example of this is when reading a Bluetooth connection – this is a blocking method, i.e. it waits until it has something to read. Separate threads were created for the Bluetooth processes to avoid freezing the UI thread.

When the Bluetooth connection does receive something to read, Broadcasts [44] and LocalBroadcastManager [45] were used to alert any active activities that were listening for that broadcast and then act accordingly. An example of this is the robot running message: this is shown while the robot is running and stops when the robot replies.

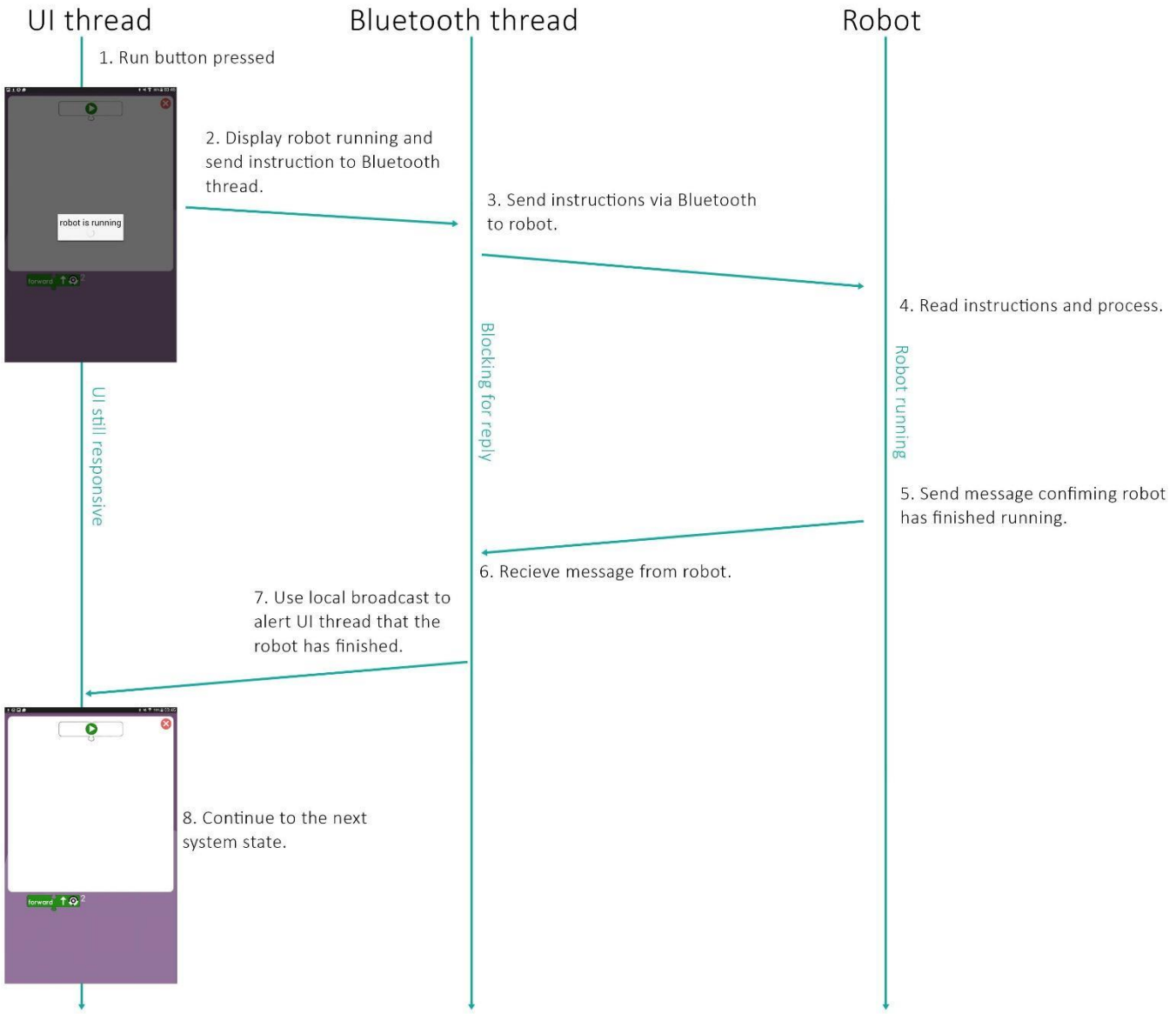

Figure 62: Demonstration of concurrent thread use.

6.1.6 Transferring and sharing data between activities

In Android, to launch one activity from another an Intent [46] is used. Data can be packaged with the intent in key value pairs, which can be extracted in the newly created activity using the key to get the value. However, if the data being passed is required in several system-wide activities, this can become quite computationally expensive. For this reason the application class [47] (which is the base class of Android applications) was extended to store a reference to the Bluetooth connection. A single connection could then be accessed system-wide without having to be packaged as part of the intent every time.

6.1.7 Final app screens

Below are a selection of screen captures from the final app.

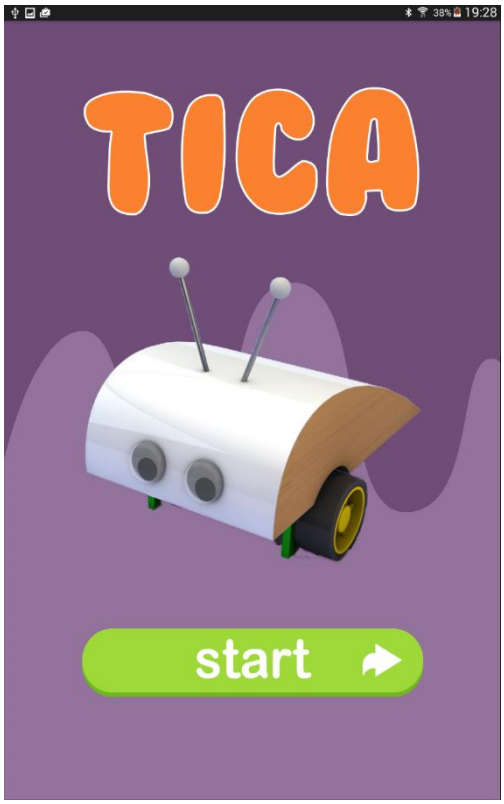

Figure 63: Splash screen.

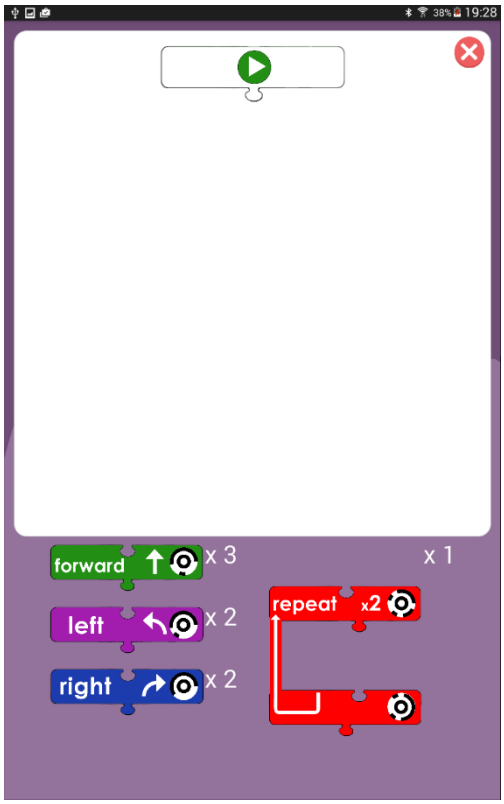

Figure 64: Task screen interface.

Figure 65: Task tangible interface.

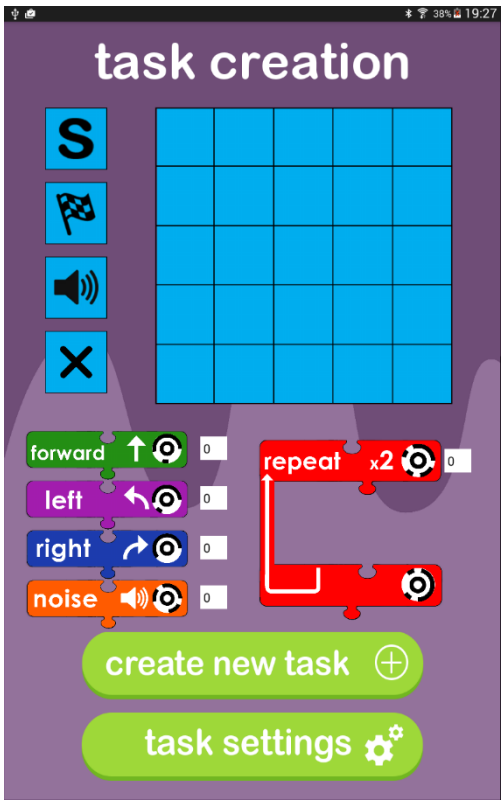

Figure 66: Task creation

6.2 Tangible Implementation

Using the 3D models from the design stage, the blocks were printed and assembled.

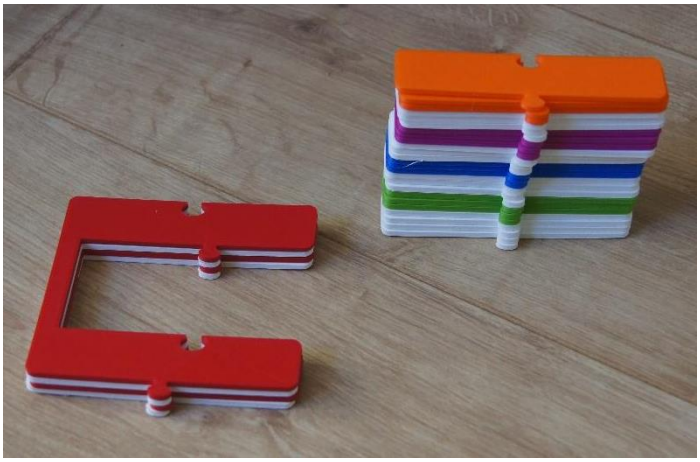

Figure 67: 3D printed instructions printed separately

Figure 68: Assembled tangible instructions

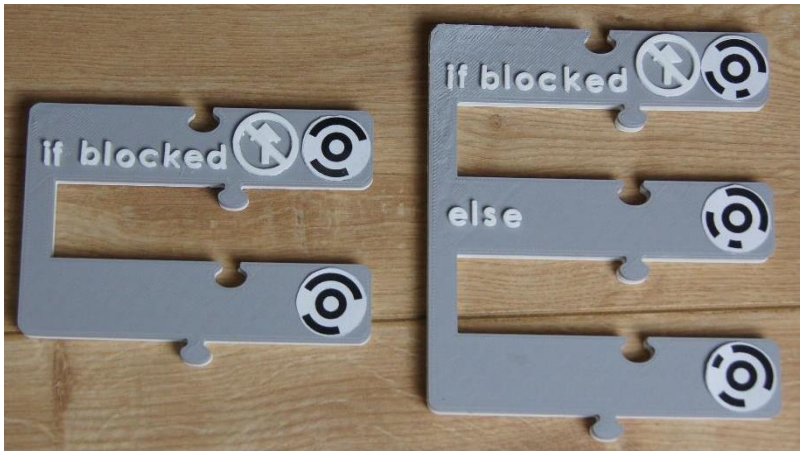

Figure 69: If and if-else instructions.

The mirror holder and stand were also printed. Basic tests were run and results showed that the angle of the stand was affecting the accuracy of the app when scanning the codes. This can be seen in the distortion of the image in Figure 70.

Figure 70: Original stand angle and resulting image with distortion of codes

To overcome this, an attempt was made to use a Google library for Optical Character Recognition [48] and a new set of tangibles was created to investigate this.

Figure 71: Tangibles for testing OCR

However, this investigation resulted in worse accuracy than the TopCode codes. As an alternative the angle of the stand was moved to be more upright (Figure 72). This rectifies the distortion effect, but unfortunately makes the angle of the tablet obstructive for users, and reduces the number of instructions that can be used. This is a key area for future improvement.

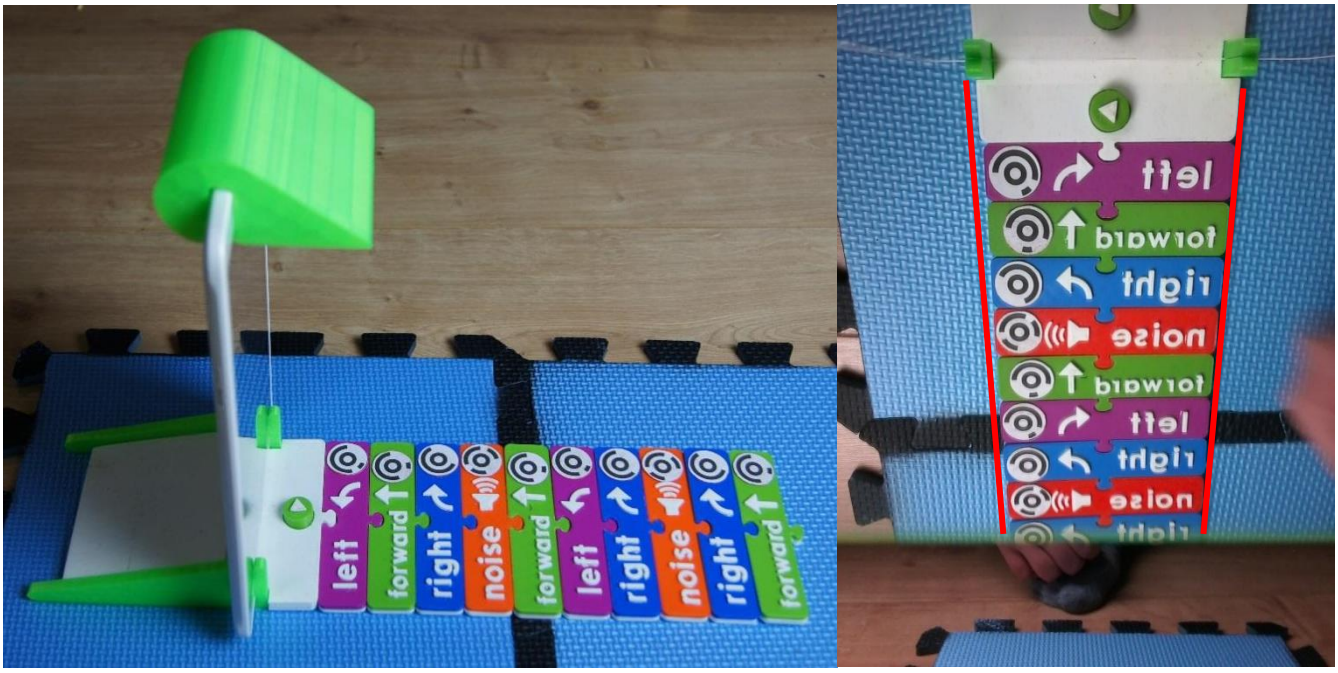

Figure 72: Upright stand angle and resulting effect on captured image.

Figure 73: Current tangible implementation

6.3 Robot implementation

The robot was built to the design specified.

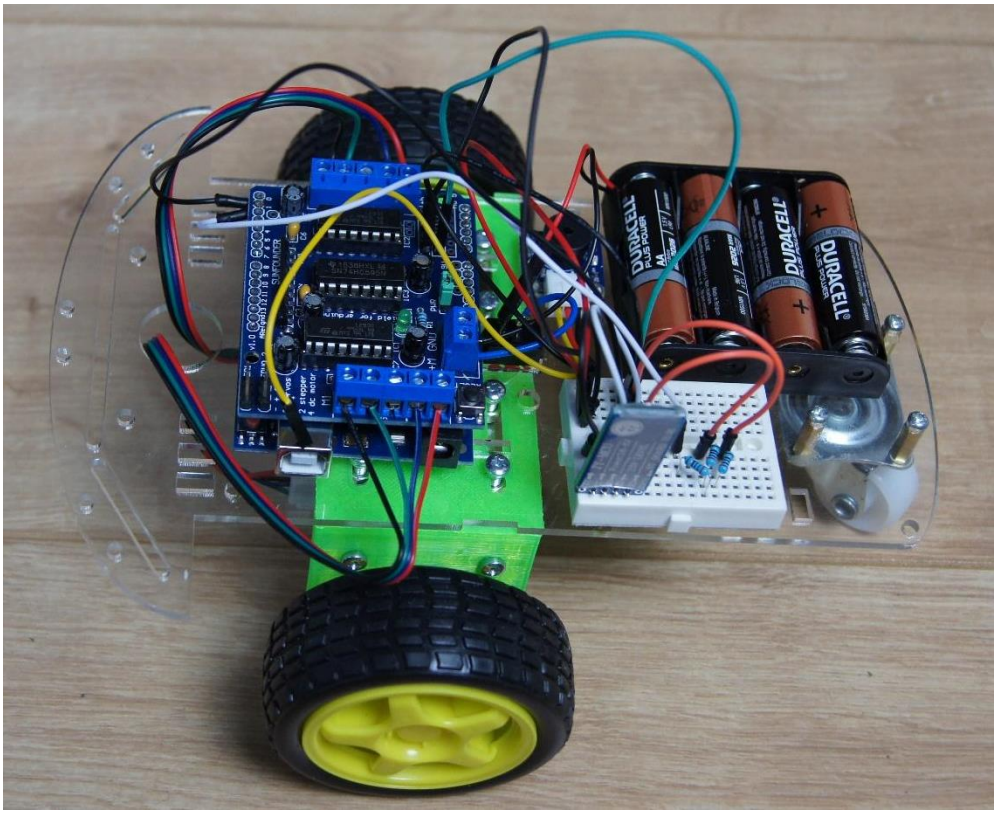

Figure 74: Robot implementation, side view.

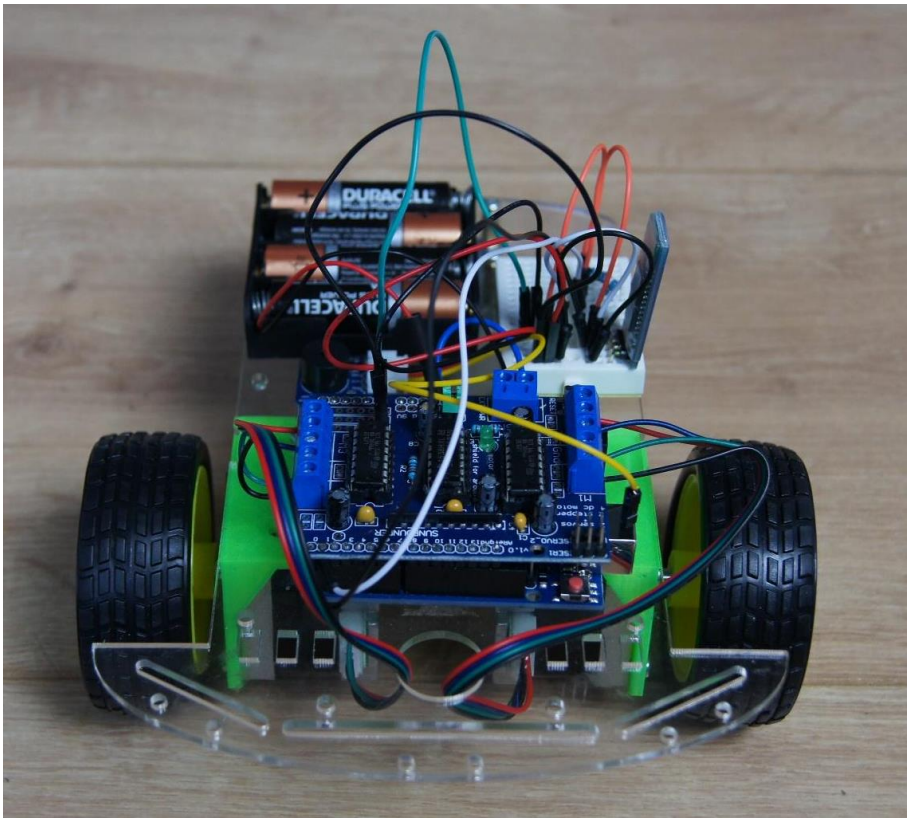

Figure 75: Robot implementation, front view.

AA batteries were insufficient, giving less than 10 minutes of run-time. A larger battery was tested to power the motors and a USB power bank for the Arduino and Bluetooth connection. This gave an improved (although still less than ideal) run-time of ~45 minutes.

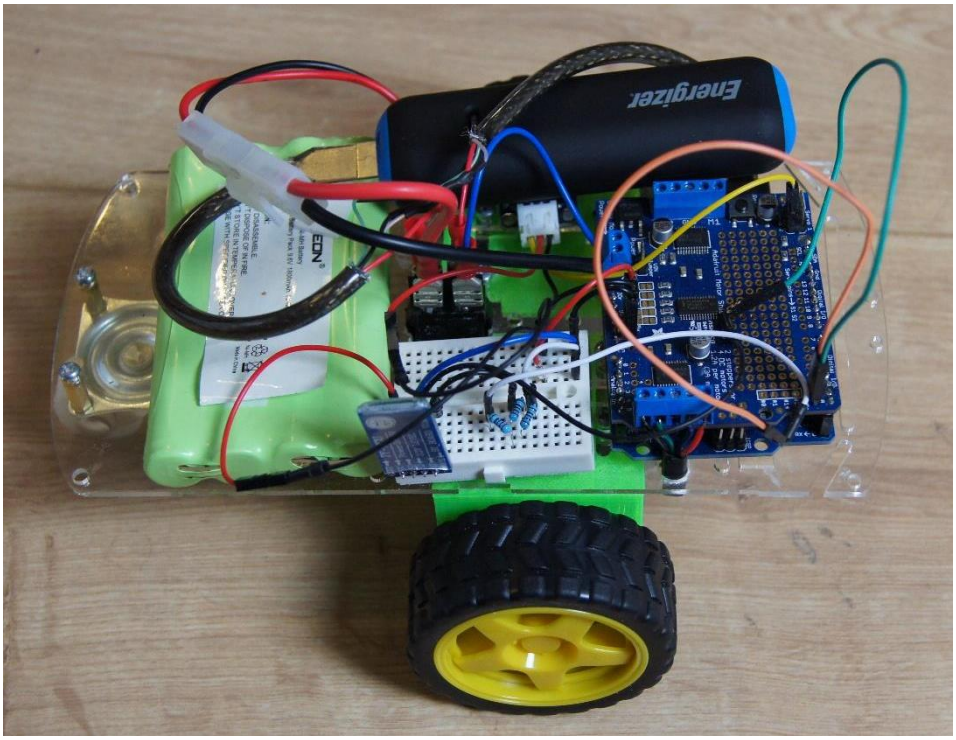

Figure 76: Robot with increased improved power.

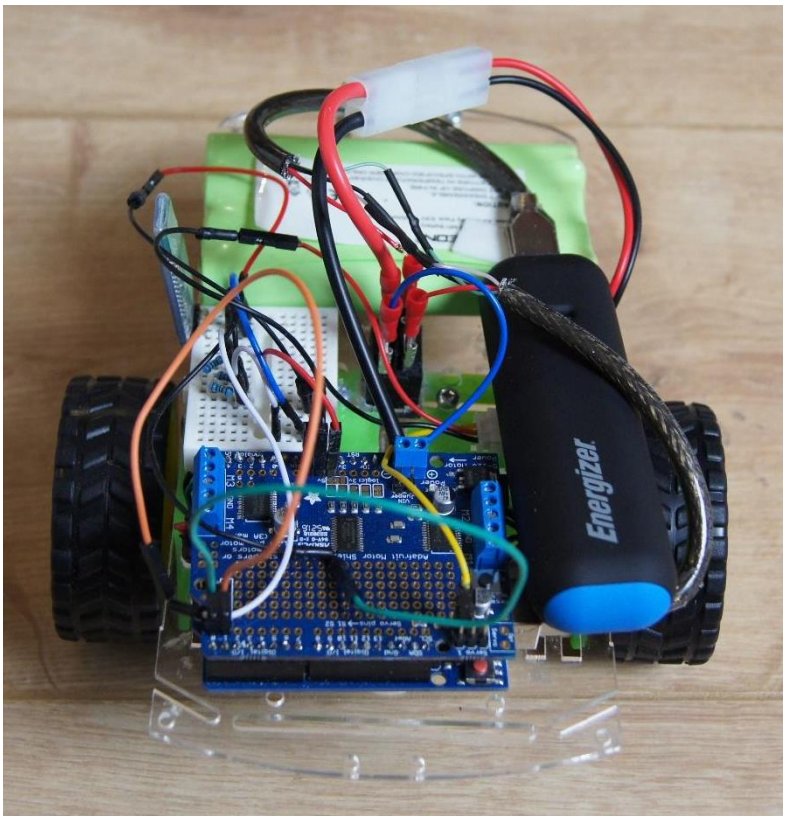

Figure 77: Robot with improved power.

Constructing the robot's case involved cutting the wooden sides using a jig-saw, then sanding them. The curved section of the robot was 3D printed, based on the design models. The case is currently white but this should be brighter in the future.

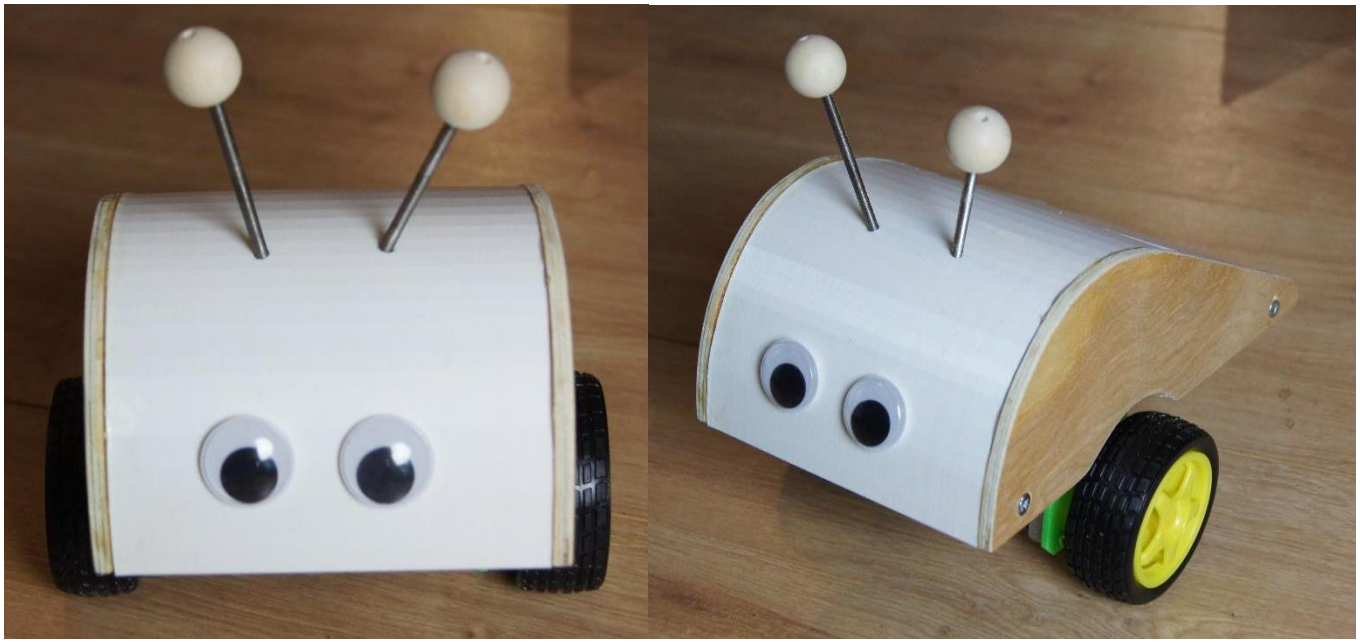

Figure 78: Robot with case

The layout tiles were painted to match the designs shown previously. The paint does not hold to the EVA foam particularly well; this is a point for future improvement. One start, one finish, four noise and twenty three empty tiles were created.

Figure 79: Painted EVA tiles

6.4 Overall system

The full system in use as a tangible and a screen interface can be seen below.

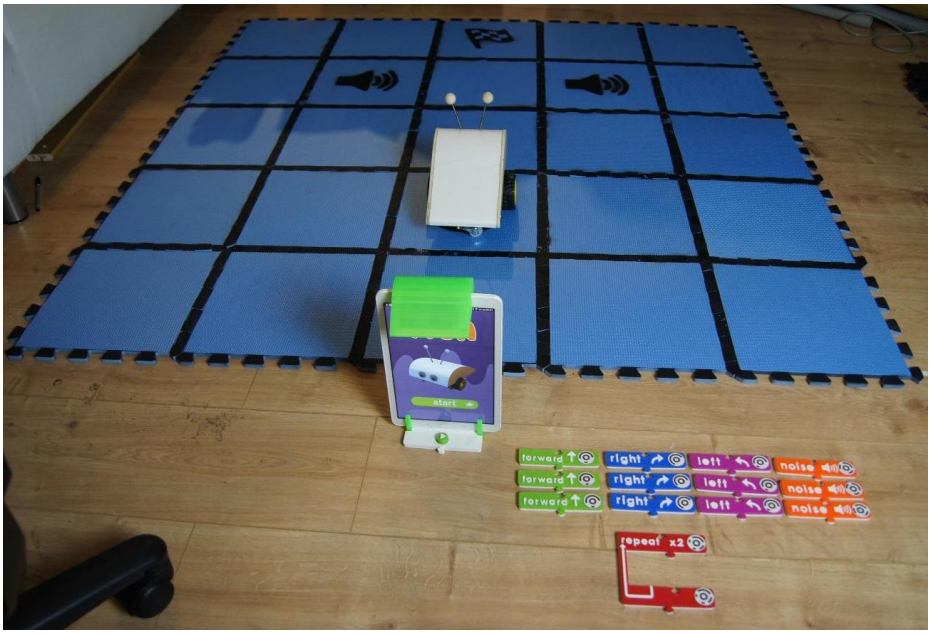

Figure 80: System set up for using tangible interface.

Figure 81: System set up for using the screen interface.

7. System testing and Evaluation techniques

7.1 System testing

The system was tested in multiple ways. As mentioned already, unit testing was used throughout development. The Espresso test recorder [49] was also used to create automated tests for the user interface. Espresso allows the recording of a test visually and then values can be asserted for elements of the display.

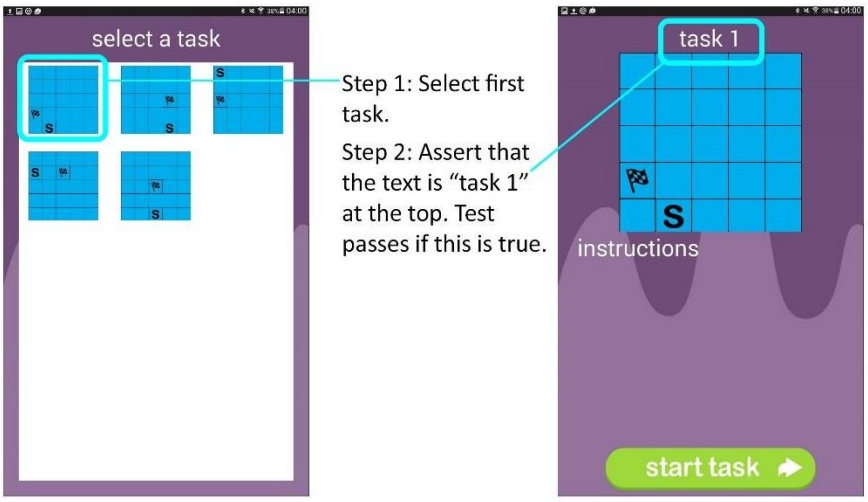

Figure 82: Example of a test created using the Espresso test recorder

During the design and development stages, a test plan was created. On implementing the system fully, the test plan was then used to test the system; the outcome is in Appendix G. These tests analysed system-wide behaviour that could not be captured via an automated system and often test one of the requirements.

User testing was carried out in conjunction with the system evaluation (see system evaluation section).

A heuristic evaluation using Nielson heuristics [50] was also conducted on the system to identify areas that could be improved (Appendix F).

7.2 Gold standard evaluation

The gold standard for an evaluation of this system would have been to run a mid- to long-term study with children in the target age group. The study would compare children taught to program using the screen interface with those taught using the tangible interface. This teaching would have happened over a number of sessions over several weeks. At the start and end of each session, pre-testing and post-testing would have been conducted. This would allow the progress of each group to be monitored over each session as well as over the whole period. The longer time period would help to investigate if the novelty of a tangible interface decreased over time. This type of evaluation was never an option for this project, due to the time and resource limitations

What was planned for this project was a smaller evaluation process. A school would be visited twice with a break of two weeks between sessions. In the first session half of the study group would use one interface, and the opposite half the other. In the second session, groups would swap over. This type of study was planned for and an ethical application created (Appendix I), but unfortunately this plan was not completed within the project timeline.

7.3 System evaluation

As an alternative, two types of testing using real users were conducted with adults. As the system had two intended users (a facilitator conducting the study and those participating in the study), user testing was done for each.

7.3.1 Aim of testing

Participant role

The evaluation had two aims:

1. To conduct real world testing of the system to allow future improvement and ensure that the data the

system recorded was accurate.

2. Using independent measures, gather preliminary data on the effects of changing the independent variable, interface type.

Facilitator role

The facilitator session was similar to aim 1 of the participant role and was designed to explore how intuitive the system was and to help improve the system in the future concerning tasks carried out by a facilitator.

7.3.2 Participants

Participant role

Participants were sourced by emailing close friends and relatives and asking them to contact people who would participate in an evaluation session lasting approximately 30 minutes, who did not have programming experience. This was a less-than-ideal selection method, but did result in 14 participants being found quickly who did not have a close personal connection to the researcher.

Facilitator role

Five participants were chosen for this role (based on Nielson's research [51]). All participants were final year computer science students at the University of Sussex. They were selected as they would likely have a similar level or computer literacy to a facilitator who would use the system to run a session. Selection was done via emailing personal contacts.

7.3.3 Materials used

Participant role

The materials used for the session were a Samsung Galaxy Tab E tablet to run the system on, the robot, tangible instructions, tiles, mirror holder and stand built during the project to be used to complete tasks. Also a stopwatch to measure time spent on tasks, paper and pen to record observations, and pre- and post- testing questionnaires (Appendix D). Questionnaires utilized advice from the Harvard university tip sheet for questionnaires [52].

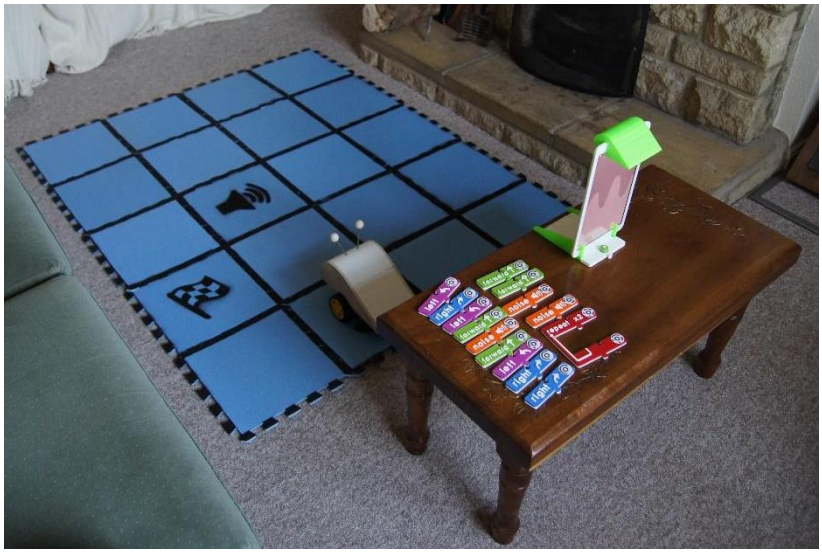

Figure 83: Session materials prepared.

Facilitator role

The same materials were used as in the participant session, except a single questionnaire was used to identify problem areas in the system (Appendix E).

7.3.4 Testing procedure

Participant role

Participants were randomly assigned one of the two interfaces to use during the session. Then each participant, one

at a time, proceeded through the following stages.

- Introduction and description of the session.
- Complete pre-test.
- Attempt 6 pre-defined tasks (identical for all participants) using assigned interface while being observed. Time spent on tasks recorded manually.
- Complete post-test and debrief.

Material for session including scripts can be found in Appendix D.

Facilitator role

Each participant, one at a time, completed the following stages:

- Introduction.
- Complete 6 pre-defined tasks based around setting up and controlling the system while being observed.
- Complete questionnaire giving feedback on experience.

Materials for session including scripts can be found in Appendix E.

7.3.5 Evaluation problems

A number of issues arose during the evaluation sessions. The sessions were conducted by a single person, therefore both session facilitation and observation were carried out by the same individual. This had a detrimental effect on the quantity and quality of the observations. If sessions were to be repeated, extra researchers would be recommended. Other issues related to technical behaviour of the system and are covered in more detail in the further improvements section.

8. Evaluation findings

8.1 Participant role findings

There were 14 participants whose ages ranged from 18 to 64 (mean age of 37.4 with a standard deviation of 17.1). All but two users (aged 61 and 63) had used a tablet prior to the evaluation.

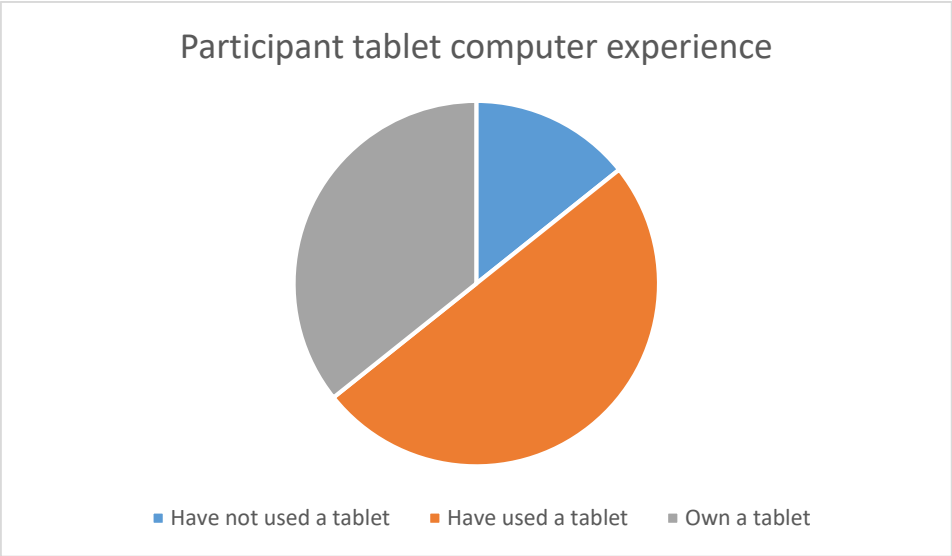

Figure 84: Participant previous tablet experience

The pre-test questionnaire involved two questions which required the participant to predict a program's behaviour. These same two questions were repeated within the post-testing, in order to see if the participant's views had changed. Two participants incorrectly answered one of these questions (one from each interface) in the pre-testing, but all answered correctly in the post-testing. Despite the statistics being too few to draw conclusions, this does indicate that those users experienced active learning during the session (but shows no difference between the interfaces). The high number of correct answers in the pre-testing indicate that, for adults, the meaning of the instructions was intuitive.

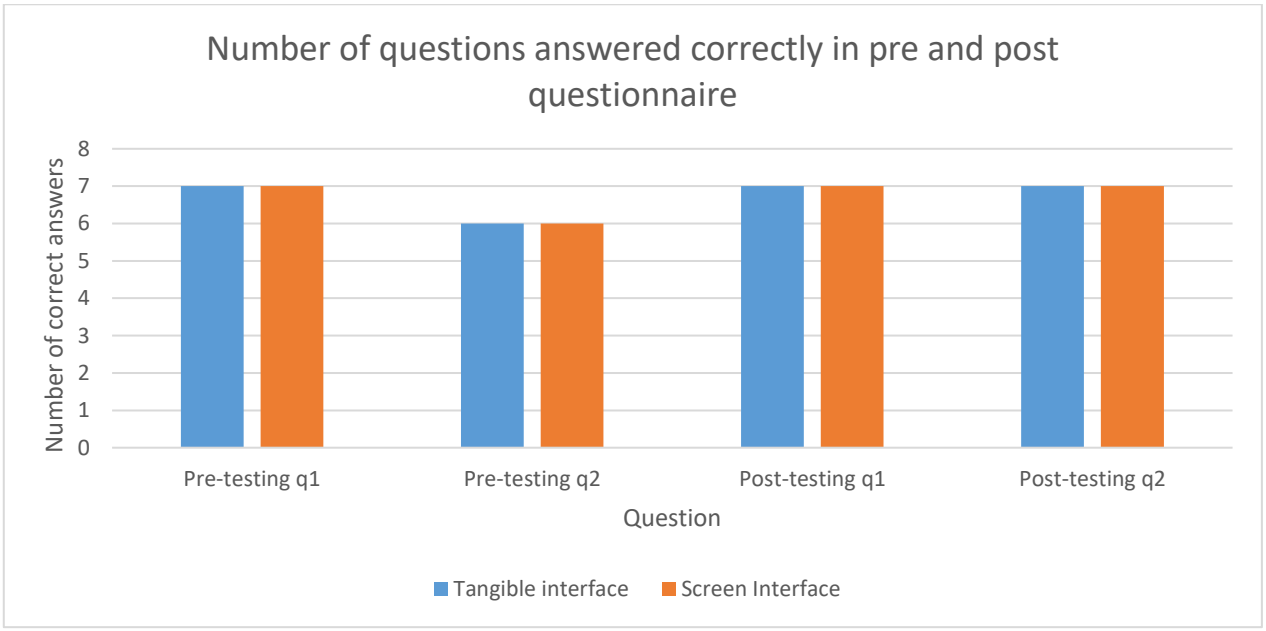

Figure 85: Number of correctly answered questions in pre and post testing

Combining all post-testing questions, 5 answers were incorrect out of 70, meaning 92.8% were correct. This would indicate that the majority of participants had a good understanding of the instructions by the end of the session. Of the incorrect answers, 2 were from users who had used the tangible interface and 3 from those using the screen

interface. This does not show a significant difference between the two interfaces.

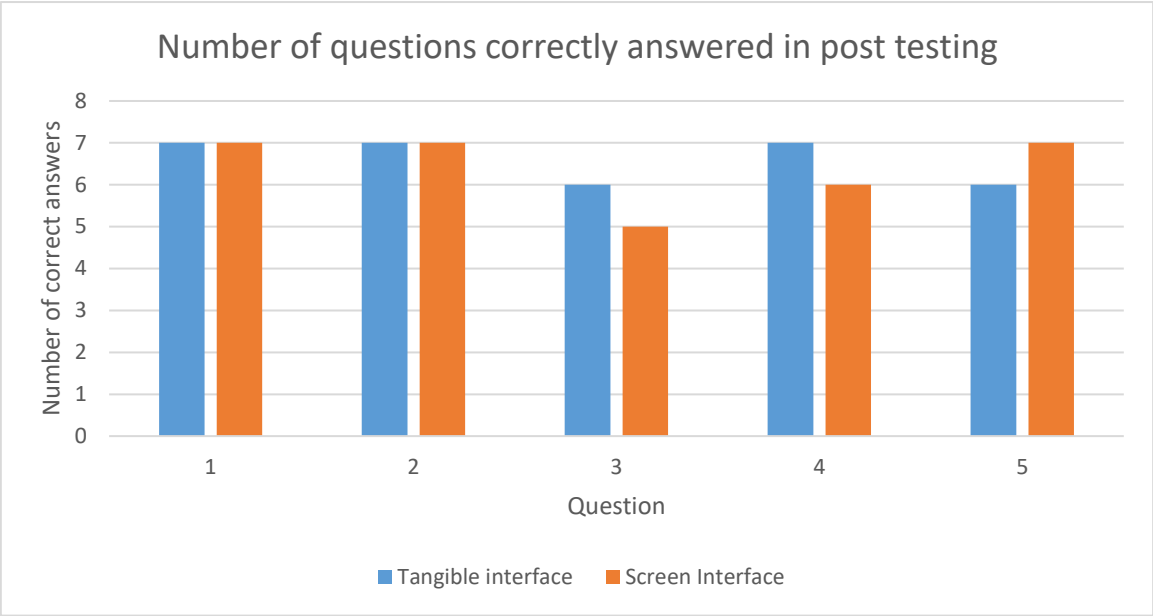

Figure 86: Number of correctly answered questions in post-testing.

All tasks were completed within two attempts regardless of the interface. Users found it easy to recognise desired and erroneous results, based on their comments when the robot behaved in a certain manner. Those participants using the tangible interface would often start to correct problems before the robot had finished the current (incorrect) attempt. In contrast this is not possible with the screen interface. The task with the most incorrect attempts was Task 2. Several participants who had an incorrect attempt commented that they expected the robot to move laterally to the right rather than turn 90 degrees to the right, when the right instruction was initiated - this area would benefit from some further consideration. Task 4 had the second highest number of incorrect attempts. This task required at a minimum 7 instructions, and was the longest program. The difficulty level may correlate with the fact that it resulted in more incorrect attempts.

The number of incorrect attempts was spread relatively evenly between the two interfaces, with the tangible interface having 4 and the screen interface 5. In general very few incorrect attempts were made - this was not surprising since the system was designed for children.

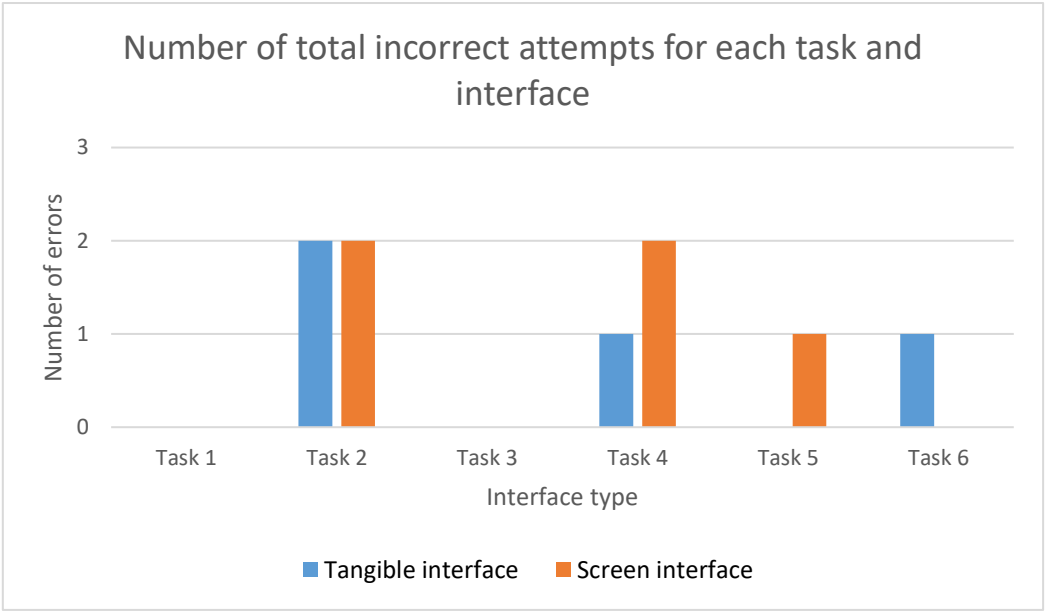

Figure 87: Number of total incorrect attempts for each task

For all tasks, the average duration was longer for the tangible interface than the screen interface. Unsurprisingly, the quickest task to complete was task 1, which only required two instructions. The slowest was task 6 which, while not requiring the most instructions, was designed to be the most complex, involving the use of a repeat as well as 3 other instruction types.

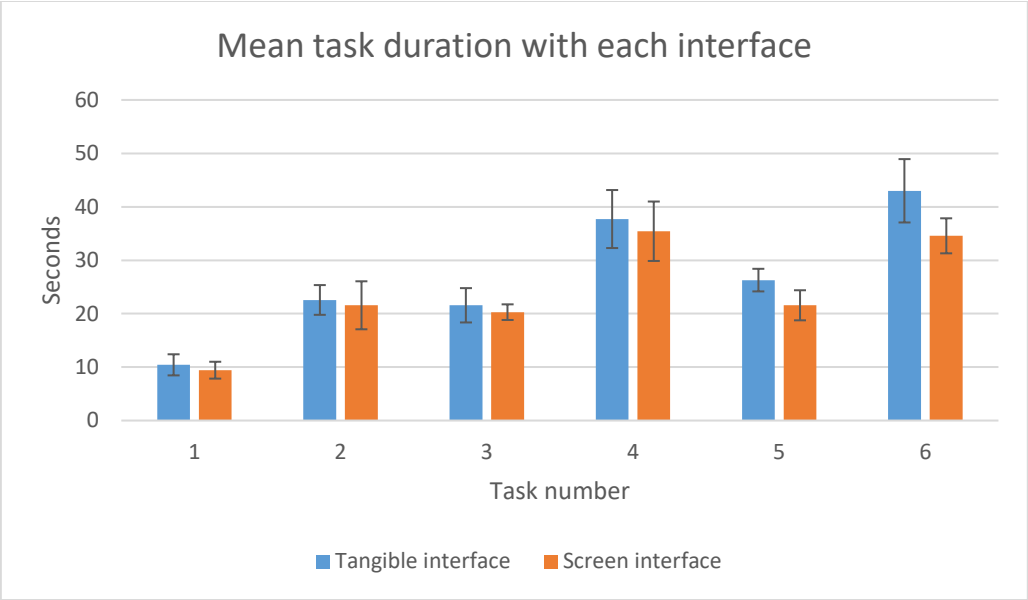

Figure 88: Mean task duration for each interface

It is interesting to note the results of the two participants (5 and 12) who had not previously used tablets. Participant 5 used the tangible interface and participant 12 the screen interface. From Figure 89 it can clearly be seen that participant 5 completed all tasks more quickly. Having just one inexperienced participant using each interface makes it impossible to draw any conclusions as to whether the interface was a factor in this difference, but nonetheless it does raise a question that would be interesting to further investigate: does previous tablet experience affect the results?

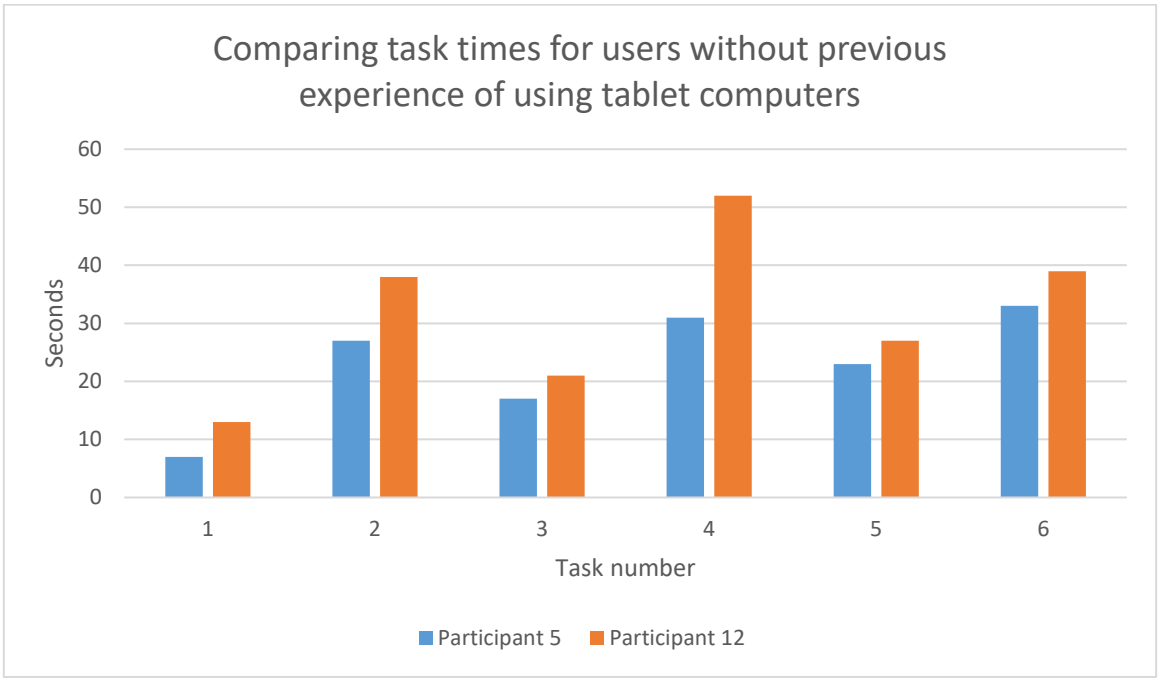

Figure 89: Comparing task times for participants 5 and 12

The difference in time between the two interfaces can further be seen when all tasks are aggregated into a single average time. The difference between the two averages is approximately 3 seconds. It should be noted that only a very small part of the session involved the participant building solutions to the problems. A lot of the time was used

setting up each task mat and with the robot running as well as the pre- and post- questionnaires and explanations.

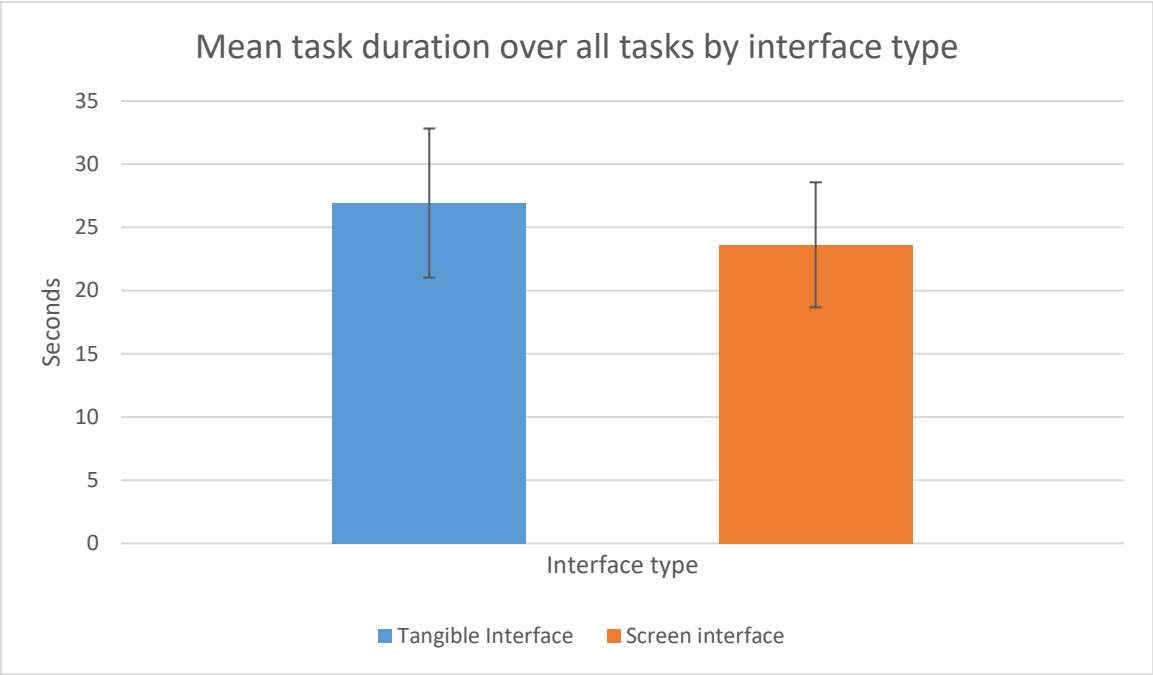

Figure 90: Average task duration over all tasks for each interface

User-reported enjoyment was overall higher than expected, considering study participants were not the target user group. Enjoyment ratings were higher for the tangible interface than the screen interface. This may be in part due to the fact that the tangible interface was a novel interface that participants had not seen before. Research over a longer time period would shed light on whether enjoyment of the tangible interface dropped more quickly than that of the screen interface. Another consideration is the subjectivity involved within users' enjoyment ratings.

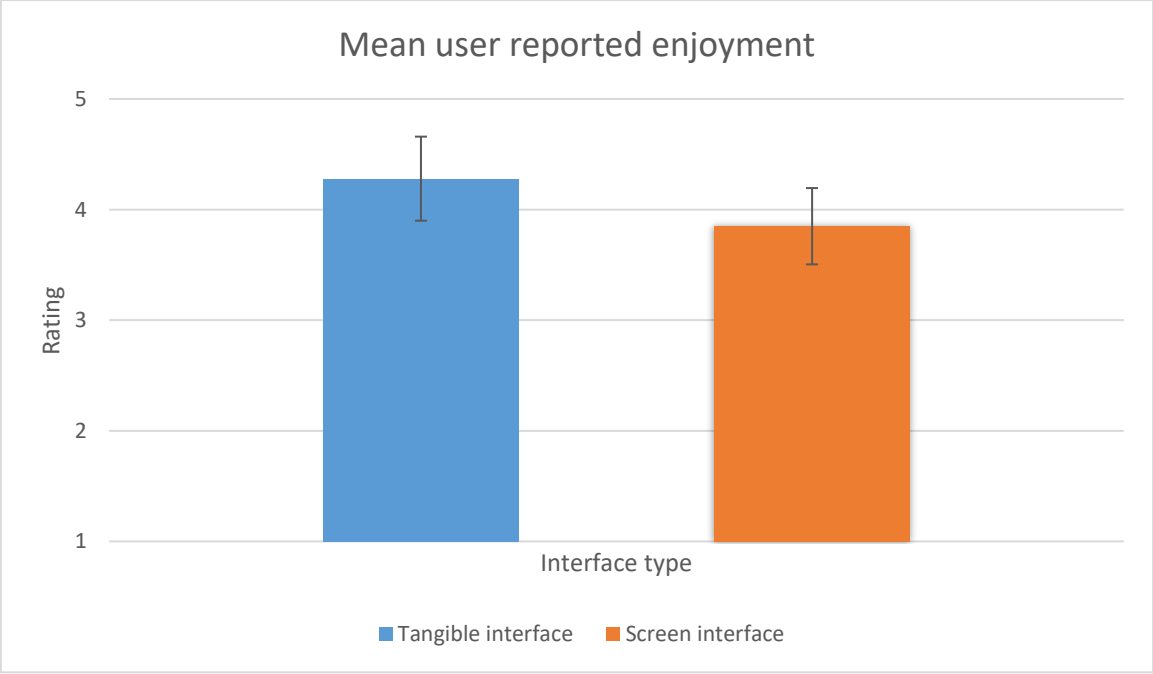

Figure 91: Average user reported enjoyment ratings

8.2 Facilitator role findings

Time taken to complete facilitator tasks varied, with Task 1 (connecting the robot and tangible system to the app via Bluetooth) taking the longest (average 59.4s). All tasks took on average under 60 seconds for people who had not previously used the system, suggesting it is relatively easy to use and intuitive.

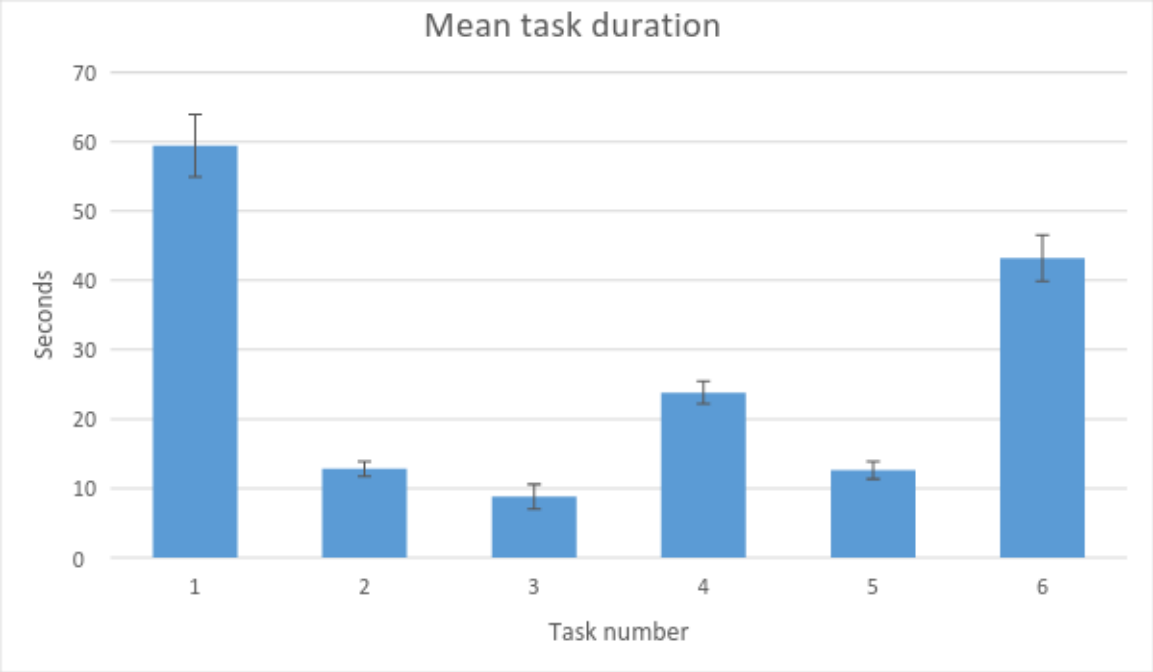

Figure 92: Average task duration

Task 1 was found to be the most confusing by all users. This was not surprising, as task one required turning on two devices. Confusion over whether the robot and tangible system were turned on stemmed from the fact there is no clear indicator of this - a good point for future improvement.

8.3 Future improvements

Based on the feedback from both evaluation sessions, the observations taken and the issues that arose, and the heuristic evaluation, a number of improvements to the system have been identified. A brief overview is given below.

8.3.1 Robot improvements

Currently it is very hard to see if the robot and the tangibles are turned on and connected to the application. The state of these two devices could be made clearer by adding an LED to both in a visible place and an icon in the app showing connection.

The battery life of the robot is currently approximately 45 minutes, meaning that multiple batteries are needed to run a single session, and time is required to replace the batteries regularly. This issue is compounded by the fact the robot does not signal when the battery is low. This resulted in some erratic behaviour during the evaluation sessions when the batteries were running low. In addition to a better power source, which would allow the robot to run for longer, some form of audible or visual alert should be given before the low batteries affect performance.

Currently the case is attached to the robot via four screws. This results in requiring a screwdriver to change batteries which takes some time. A better system could be found for attaching the case.

The back wheel on the robot sometimes does not swivel to be in line with the robot. As a result, the robot sometimes goes off-line when moving forward, due to increased friction on one side. A better wheel assembly would improve this.

During extended periods of use, the robot's stepper motors overheat. This reduces the lifetime of the motors and causes a potential hazard for children. This could be reduced by using motors with a higher voltage.

One possible alternative to the bespoke robot would be using an off-the-shelf robot or kit. One possible example is the WowWee MiP [53], a self-balancing robot that has an Android development kit available which works over Bluetooth. This could be a relatively simple way to solve a number of the issues with the system.

Figure 93: WowWee MiP [53]

8.3.2 Language improvements

The repeat instruction is currently quite restrictive in both the number of instructions it can contain as well as the number or iterations it provides. While the "if" instruction has been implemented, it is hard to create tasks that require the use of the "if" instructions. A bigger repeat instruction would allow a more natural use of the "if" instruction.

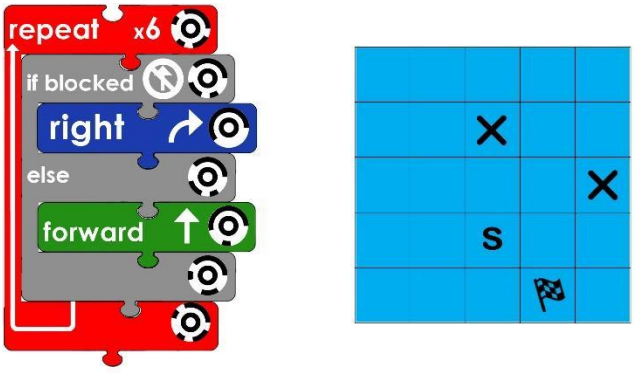

Figure 94: Potential task using a larger repeat instruction.

One solution would be to build a repeat instruction that could be expanded and collapsed. A possible design in shown in Figure 95.

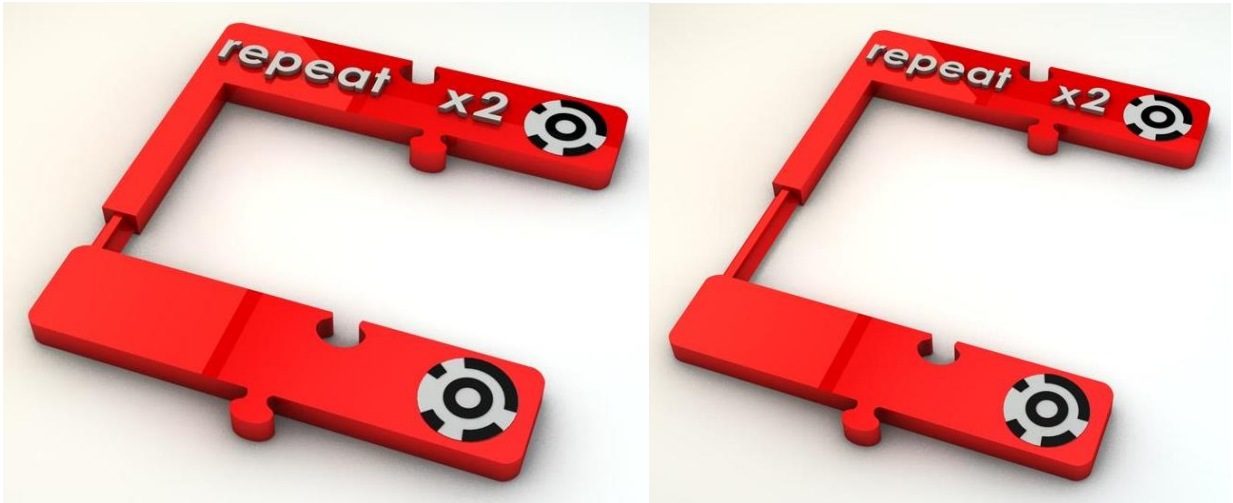

Figure 95: Variable size repeat instruction.

However, it should be noted that as more instructions are added, the differences between the two interfaces increases. This is due to the fact that the screen interface has a limited area to display available instructions, so

either instructions have to be reduced in size or a scrolling system implemented, meaning that the user cannot see all available instructions at once. This is not an issue for the tangible interface.

Currently the left and right instructions turn the robot 90 degrees relative to the direction it is facing. This caused some confusion during evaluation. A possible alternative would be for the robot to travel in the direction of the instruction with respect to the layout irrespective of the direction the robot is facing. This would have been an ideal concept to test through user centered design.

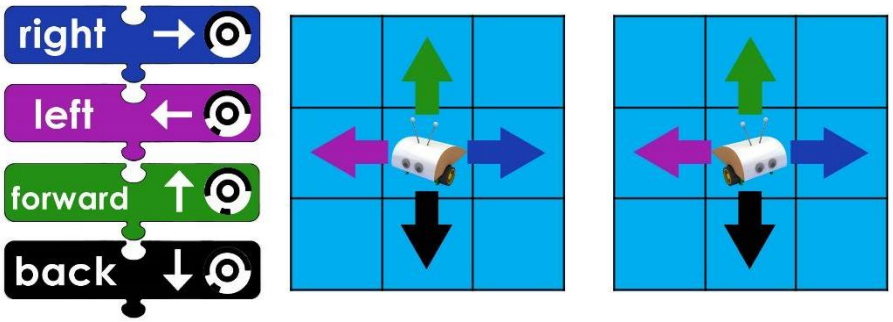

Figure 96: Possible alternative direction instructions.

8.3.3 App improvements

During evaluation it was noted that a participant had started solving one of the problems while the mat was being set up. This would then result in a quicker time for a solution so worth bearing aware of and possibly designing to overcome.

Currently task layouts are on a 5 x 5 grid; it would be good for this to be adjustable. In this way, smaller layouts could be used for simpler tasks or where the available floor area is limited.

Currently tasks cannot be deleted from the app. A facilitator may want to remove some tasks from the system, to control the number to choose from. It would also be useful to be able to export all the performance data into a document rather than having to manually record it.

8.3.4 Stand and tile improvements

When in the stand, due to the upright angle of the tablet, the viewing angle of the screen is not accommodating to the user. As explained in the design section, this angle was used to increase the accuracy when scanning the TopCodes. While a small unsuccessful test was run with text recognition, this is a key area that should be revisited. This would then allow for the angle of the stand to be altered.

The tiles created for the layouts have become worn; a robust alternative would be an improvement.

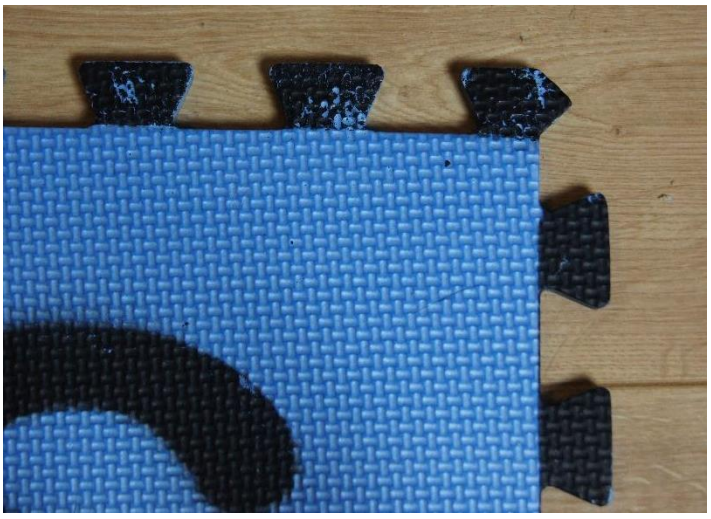

Figure 97: Paint falling off tiles.

There is an occasional issue that that when using the tangible system, if the user keeps their hand on the play button, it can obscure some of the codes in the picture captured. Currently this is overcome by telling the user to remove their hand after pressing the button. A better method should be found.

Figure 98: Hand blocking codes in captured image.

Similarly, tangible instructions can be picked up by the system that were not intended to be if they are too close to the ones intended to be scanned. This could be solved by adding a mat to the tangible system that shows the active area that is scanned.

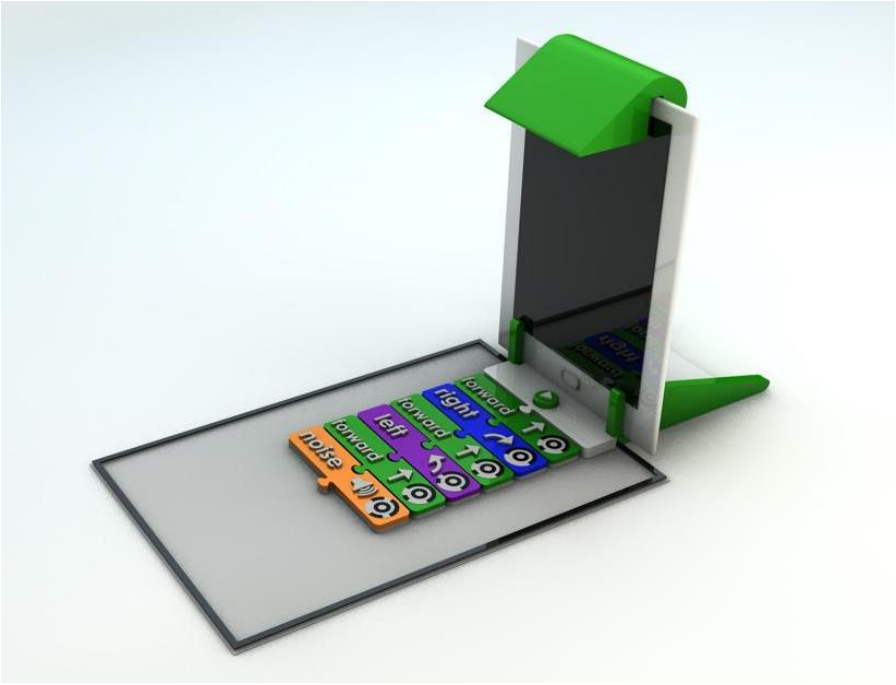

Figure 99: System with mat.

9. Conclusion

9.1 Project conclusion

Overall the project has been successful. It has fulfilled the primary aims of the project by creating a system that could be used to compare screen and tangible interfaces for programming. The secondary objective to create an "if" instruction has been implemented, but requires changes in the system to be truly useful. While a small evaluation was run, it was not run with the target user group and as a result has not managed to gain data that could be used to indicate an answer to the question set in the title, but this was always an ambitious target. Currently the weakest part of the system is the robot. I do feel that if the major issues with the robot were solved, the system created is in a position where it could start to be used to investigate the title question. A number of improvements that could be made to the system to increase its effectiveness and usability have been highlighted.

The limited data gathered indicated that, with adults, the screen interface was quicker to use, but that users enjoyed the tangible interface more.

The system was harder to implement than initially thought, but this was a good learning experience in adapting to the circumstances as they change, and in time-management. I have learnt an extensive amount by working on this project. Through the background reading I learned about the general area of tangible programming and its history. I also learnt about the complications involved when potentially working with children, and the extra ethical considerations required. This project allowed me to put into practice skills I had previously learnt but had not used; for example I had never had a database (databases course) as part of a system. This project also allowed me to practice using a three tier architecture (software engineering course). I now have a basic understanding of the Arduino system and this knowledge is already being used in a separate project. Running the evaluation was a very useful experience and reinforced the difficulties in running a fair and well managed session. Finally, I have learnt a huge amount about the Android framework and working within it.

If I were to repeat this project, the most significant change I would make would be to apply for ethical review earlier to allow a higher chance of success. I would also consider using a pre-made robot, as this was the area I have struggled with most, and removing this step would have allowed more time to work on the other components of the system. I would continue to use a visual recognition system for the tangible instructions, as I feel this was the right design choice and worked well during evaluation. I would also stick with a robot as the output of the system as the users reported enjoying this and found it easy to identify the issue when instructions were not correct.

9.2 Further study

The obvious area for further study would be to run the evaluation with the target user group (ideally a group of at least 60 children aged 5-7 years).

In developing the system and in the small evaluation study, a number of further research questions arose, including:

- If tangible interfaces demonstrate an advantage, is there an age when this advantage drops off?
- How does the number of users using the system at one time affect the performance of each interface?
- What effect does the amount of previous experience a user has with tablet computers have on the results?
- If tangible interfaces demonstrate an advantage, is there a point in program complexity that this advantage diminishes? (Currently the system is limited to the number of instructions that can be used at once.)
- When tasks can be completed in multiple ways, are there patterns that emerge in the solutions given using a tangible interface versus a screen interface?
- This project was based on the difference between input interfaces. What effect does the output modality have? Currently the output is a physical robot, but does the effectiveness of the input change with output modality? For example, is a screen input interface and screen output a better combination than tangible input interface and screen output interface etc.

10. References

[1] Peyton-Jones, S., B. Mitchell, and S. Humphreys. (2013). Computing at school in the UK: from guerrilla to gorilla. *Communications of the ACM*.

[2] Department for Education (2013) *The National Curriculum in England: computing programmes of study.* [Online] Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study [Last accessed 03/10/2016].

[3] Horn, M.S. and Jacob, R.J., (2007). Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (pp. 159-162). ACM.

[4] Ofcom. (2014). *One in three children has their own tablet computer.* [Online] Available: https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2014/media-lit-audit-oct2014. [Last accessed 04/10/2016].

[5] besa. (2015). *Tablet adoption continues to rise; barriers to adoption shift.* [Online] Available: http://www.besa.org.uk/news/besa-press-release-tablet-adoption-continues-rise-barriers-adoption-shift/. [Last accessed 05/10/2016].

[6] Catalina Naranjo-Bock. (2011). *Effective Use of Color and Graohics in Applications for Children, Part 1: Toddlers and Preschoolers.*[Online] Available: http://www.uxmatters.com/mt/archives/2011/10/effective-use-of-color-and-graphics-in-applications-for-children-part-i-toddlers-and-preschoolers.php. [Last accessed 15/10/2016].

[7] Strizer, I. (2015). *Typography for Children.* [Online] Available: https://www.fonts.com/content/learning/fyti/situational-typography/typography-for-children. Last accessed 16/03/2017.

[8] Debra Gelman. (2014). *Design for Kids: Digital Products for Playing and Learning.*[Online] Available: http://www.uxmatters.com/mt/archives/2014/07/design-for-kids-digital-products-for-playing-and-learning.php. [Last accessed 15/10/2016].

[9] Kid Sense. (2014). *Fine Motor Development Chart.* [Online] Available: http://www.childdevelopment.com.au/home/183. [Last accessed 29/10/2016].

[10] Xie, L., Antle, A.N. and Motamedi, N., (2008). Are tangibles more fun?: comparing children's enjoyment and engagement using physical, graphical and tangible user interfaces. In *Proceedings of the 2nd international conference on Tangible and embedded interaction* (pp. 191-198). ACM.

[11] Horn, M.S., Solovey, E.T., Crouser, R.J. and Jacob, R.J., (2009). Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 975-984). ACM.

[12] Sapounidis, T. and Demetriadis, S., (2013). Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. *Personal and ubiquitous computing*, *17*(8), pp.1775-1786.

[13] Sapounidis, T., Demetriadis, S. and Stamelos, I., (2015). Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing*, *19*(1), pp.225-237.

[14] McNerney, T.S., (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing*, *8*(5), pp.326-337.

[15] Logo foundation. (2010). *A Logo Primer.* [Online] Available: http://el.media.mit.edu/logo-foundation/what_is_logo/logo_primer.html. [Last accessed 14/0/2016].

[16] Perlman R. (1976) Using computer technology to provide a creative learning environment for preschool

children. Logo memo no 24, MIT Artificial Intelligence Laboratory Publications 260, Cambridge, Massachusetts

[17] Scratch. (2016). *Create stories, games, and animations.* [Online] Available: https://scratch.mit.edu/. [Last accessed 28/10/2016].

[18] Scratch. (2016). *Scratch statistics.* [Online] Available: https://scratch.mit.edu/statistics/. [Last accessed 17/10/2016].

[19] Scratch. (2016). *For Parents.* Available: https://scratch.mit.edu/parents/. Last accessed 19/04/2017.

[20] Google for Education. (2016). *A library for building visual programming editors.* [Online] Available: https://developers.google.com/blockly/. [Last accessed 18/10/2016].

[21] Lightbot. (2016). *Solve puzzles using programming logic.* [Online] Available: https://lightbot.com/. [Last accessed 18/10/2016].

[22] Osmo. (2016). *Osmo Coding.* [Online] Available: https://www.playosmo.com/en/coding/. [Last accessed 19/10/2016].

[23] Hu, F., Zekelman, A., Horn, M. and Judd, F., (2015). June. Strawbies: explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 410-413). ACM.

[24] Michael Horn. (2015). *Tangible Object Placement Codes.* [Online] Available: http://users.eecs.northwestern.edu/~mhorn/topcodes/. [Last accessed 28/10/2016].

[25] Tufts University HCI lab. (2013). *tern tangible programming.* [Online] Available: http://hci.cs.tufts.edu/tern/. [Last accessed 26/10/2016].

[26] Horn M.S., Jacob R.J.K. (2014). *Tangible Programming in the Classroom with Quetzal.* [Online] Available: http://engineering.tufts.edu/gradresearch/documents/GradResearch2006/horn.pdf. [Last accessed 22/10/2016].

[27] Primo toys. (2016). *Teach your child to code before they can read.* [Online] Available: https://www.primotoys.com/. [Last accessed 25/10/2016].

[28] Google. (2016). *Project Bloks.* [Online] Available: https://projectbloks.withgoogle.com/. [Last accessed 28/10/2016].

[29] legislation.gov.uk. (1998). *Data Protection Act 1998.* [Online] Available: http://www.legislation.gov.uk/ukpga/1998/29/contents. [Last accessed 28/10/2016].

[30] Android. (2016). *Bluetooth API guide.* [Online] Available: https://developer.android.com/guide/topics/connectivity/bluetooth.html. [Last accessed 1/11/2016].

[31] Bluetooth. (2015). *Bluetooth Core Specifications 4.2.* [Online] Available: https://www.bluetooth.com/specifications/adopted-specifications. [Last accessed 1/11/2016].

[32] Codd, E.F., 1974. *Recent Investigations in Relational Data Base Systems* (pp. 1017-21). IBM Thomas J. Watson Research Division.

[33] SQLite. (2016). *About SQLite.* [Online] Available: https://www.sqlite.org/about.html. Last accessed 8/3/2017.

[34] Norman, D (1988). The Design of Everyday Things. New York: Basic Books.

[35] Budiu, R. (2014). *Memory Recognition and Recall in User Interfaces.* [Online] Available: https://www.nngroup.com/articles/recognition-and-recall/. Last accessed 17/2/2017.

[36] Wikipedia. (2017). *PID controller.* [Online] Available: https://en.wikipedia.org/wiki/PID_controller. Last accessed 22/11/2017.

[37] git. (2017). *distributed is the new centralized.* [Online] Available: https://git-scm.com/. Last accessed 21/3/2017.

[38] GitHub. (2017). *The World's leading software development platform.* [Online] Available: https://github.com/. Last accessed 20/3/2017.

[39] Android. (2016). *Code Style for Contributors.* [Online] Available: http://source.android.com/source/code-style.html. Last accessed 3/3/2017.

[40] Android. (2016). *Activities API Guides.* [Online] Available: https://developer.android.com/guide/components/activities/index.html. Last accessed 11/03/2017.

[41] Android. (2016). *Fragments API Guide.* [Online] Available: https://developer.android.com/guide/components/fragments.html. Last accessed 11/03/2017.

[42] Android. (2016). *Drag and Drop.* [Online] Available: https://developer.android.com/guide/topics/ui/drag-drop.html. Last accessed 12/3/2017.

[43] Android. (2016). *OnClickListener API.* [Online] Available: https://developer.android.com/reference/android/view/View.OnClickListener.html. Last accessed 12/03/2017.

[44] Android. (2016). *Broadcasts API Guide.* [Online] Available: https://developer.android.com/guide/components/broadcasts.html. Last accessed 15/6/2017.

[45] Android. (2016). *LocalBroadcastManager API.* [Online] Available: https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html. Last accessed 15/03/2017.

[46] Android. (2016). *Intent API.* [Online] Available: https://developer.android.com/reference/android/content/Intent.html. Last accessed 18/03/2017.

[47] Android. (2016). *Application API.* [Online] Available: https://developer.android.com/reference/android/app/Application.html. Last accessed 19/03/2017.

[48] Google. (2016). *Text Recognition API Overview.* [Online] Available: https://developers.google.com/vision/text-overview. Last accessed 28/01/2017.

[49] Android Studio. (2017). *Create UI Tests with Espresso Test Recorder.* [Online] Available: https://developer.android.com/studio/test/espresso-test-recorder.html. Last accessed 17/03/2017.

[50] Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces, *Proc. ACM CHI'90 Conf.* (Seattle, WA, 1-5 April), 249-256.

[51] Nielsen, J. (2000). *Why You Only Need to Test with 5 Users.* [Online] Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. Last accessed 16/03/2017.

[52] Harrison, C. (2007). *Tip Sheet on Question Wording.* [Online] Available: http://psr.iq.harvard.edu/files/psr/files/PSRQuestionnaireTipSheet_0.pdf. Last accessed 20/03/2017.

[53] WowWee. (2016). *MiP*. [Online] Available: http://wowwee.com/mip/. Last accessed 8/04/2017.