Turtlebot Software for Schools Outreach

Cate Fitzpatrick

Department of Computer Science

Aberystwyth University

Wales

September

2025

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science

Degree: MSc Advanced Computer Science

Module: Dissertation (CSM9060)

Supervisor: Dr Neal Snooke

Abstract

The abstract stands alone as a very short version of the dissertation.

The abstract should state the scope and principal objectives of the project, describe the methods, summarize the results and state the principal conclusions.

**(Max. 500 words.)**

Declaration of originality

I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work, I understand and agree to abide by the University’s regulations governing these issues.

**Name:** Cate Fitzpatrick   
**Date:** dd/mm/yy

Consent to share this work

* By including my name below, I hereby agree to this thesis being made available to other students and academic staff of the Department of Computer Science, Aberystwyth University.

**Name:** Cate Fitzpatrick   
**Date:** dd/mm/yy

Acknowledgement

To whoever has the patience to read this :-)

This section is customary, but not obligatory. It makes a brief statement of thanks to those who have helped.

Table of Contents

[Chapter 1. Introduction 1](#_Toc205298679)

[Chapter 2. Literature review 2](#_Toc205298680)

[Chapter 3. Reporting on the project – the core chapters 3](#_Toc205298681)

[Chapter 4. Requirements 4](#_Toc205298682)

[Chapter 5. Spikework 5](#_Toc205298683)

[Picking software 5](#_Toc205298684)

[Java 5](#_Toc205298685)

[Python 5](#_Toc205298686)

[Conclusion 6](#_Toc205298687)

[Executing files 6](#_Toc205298688)

[The subprocess module 7](#_Toc205298689)

[The os module 7](#_Toc205298690)

[The exec function 7](#_Toc205298691)

[Threading 8](#_Toc205298692)

[Syntax highlighting 8](#_Toc205298693)

[HTML pages 8](#_Toc205298694)

[tkhtmlview 9](#_Toc205298695)

[Webview 9](#_Toc205298696)

[Conclusion 9](#_Toc205298697)

[Chapter 6. Design 10](#_Toc205298698)

[d 10](#_Toc205298699)

[Chapter 7. Implementation 12](#_Toc205298700)

[Chapter 8. Testing 13](#_Toc205298701)

[Chapter 9. Packaging? 14](#_Toc205298702)

[Chapter 10. Critical Evaluation 15](#_Toc205298703)

[Chapter 11. Conclusion 16](#_Toc205298704)

[References 17](#_Toc205298705)

[Appendix A 18](#_Toc205298706)

[Appendix B- Manual testing table 19](#_Toc205298707)

List of Figures

[Figure 1 Structure of a dynamic website (Edel Sherratt) 5](#_Toc205555859)

[Figure 2- Setup wizard flow chart 13](#_Toc205555860)

1. Introduction

Background to the project, motivation, leading to project aims and objectives.

* What problem was tackled?
* Why was that problem tackled?
* How (in outline) was the problem tackled?
* Clear statement of project aims and objectives.
* Guide to subsequent chapters.

1. Literature review

## Introduction

The use of robots to help people develop their coding skills has been a long-standing practice spanning over 50 years. A large quantity of these robots have had options to draw lines and shapes. Although the aims and base function of these systems have not changed drastically, the hardware and software have developed over time to provide a better user experience. Examples of this are the addition of graphical tools and hardware parts becoming smaller and cheaper. The first Turtles were developed as part of the development of LOGO [A]. Many of the more modern versions, software and hardware, have stemmed from this, such as Python’s Turtle Graphics [B].

This literature will explore the history of Logo and turtles, as well as other systems with the same aim.

## The early history of Logo and Turtles

### Logo

Logo was first invented in 1966 by Seymour Papert, Wally Feurzeig and Cynthia Solomon. In 1966, Papert visited multiple classes using programming languages and decided that a programming language designed specifically for children was required [C]. This language was then made and used in labs within the same year [B]. By 1967 the language was being tried out in schools.

Logo is a dialect of Lisp with some changes. One of the most important change was the introduction of commands[B]. Before this there were only procedures that could not return values. By introducing the option to return values, the options for programs greatly increased. Other smaller changes were made to make the language more child friendly. Lisp uses many parentheses, putting them around everything. Logo removed these.

Another choice taken to make Logo simpler is that it does not have an IDE. The idea behind this was so that the focus was to learn the language rather than trying to learn an IDE as well[B].

The first Logo programs were used for playing with words and sentences, aiming to help children to play around with them and learn grammar in the process [B]. One example of this was turning inputted words into pig Latin. These programs were originally run on time-shared computers (Digital Equipment Corporation PDP-1) [B].

The new language was seen to be successful, especially when the Logo environment ideology was kept to, which involved a culture of learning by doing [A]. The Logo group was officially formed at the MIT AI Lab in 1969[C].

### The introduction of Turtles

The idea of having physical objects to further develop the project was developed between 1969-70 [B]. Two of the objects developed were Turtles that could draw lines based on the commands they were given. The name turtle was inspired by the two automations that William Grey Walter built, which he called tortoises (Elise and Elmer more specifically)[B]. The original turtles did not have a graphical counterpart, which would come slightly later. The early turtles were connected to the computer via wiring to receive commands and send feedback[F].

The turtles had six commands: FORWARD, BACKWARD, RIGHT, LEFT, PENUP and PENDOWN. These could be combined within procedures to make a sequence of actions in hopes of creating shapes[A]. The layout of such a procedure to draw a triangle would be:

TO TRIANGLE

1. FORWARD 50
2. RIGHT 120
3. FORWARD 50
4. RIGHT 120
5. FORWARD 50
6. RIGHT 120

END

TRIANGLE

It was found that mistakes were more likely to lead to objective thinking as students tended to blame the turtle rather than themselves [A]. This meant that problems were dealt with constructively, focusing on why an issue occurred rather than the error itself.

### The introduction of Turtle geometry

Turtle geometry was introduced in 1970. This allowed a user to view a turtle and control it in the same manner as the Turtlebots as they worked off the same main six commands. Turtle Geometry worked with local geometry based upon the ideas of current position and heading [B]. It used standalone graphics terminals[B].

One example of its use was by Jon Pledge, a 12 year old at the time, who used it to create a universal maze-solving algorithm during an activity (now known as Pledge’s algorithm)[B].

After a conference in 1972 a company called General Turtle was formed in order to make and sell turtles[B]. Roughly three years later Marvin Minsky designed a standalone graphics system for the company [B].

### Logo since 1975

After the boom in personal computers, the development of Logo versions took up speed[B]. As of 2016 Logo has over 300 versions according to Pavel Boytchev’s Logo Tree Project[D]. Some of these are very similar to the original uses while others differed. Various programming languages implement as version of Logo by having their own version of Turtle graphics. The language was also used to create block-based learning environments such as Scratch[D][H]

## Later developments in Turtles and other robots

Similar to how there have been many versions of Logo, there have also been many different robots that be used similarly to turtles, most of which have developed from the original Logo ones. Not all of them have the ability to draw though many have it as an option, using modular sets.

The original turtles relied on a cord for communications. This was because radio links were considered to be too expensive given the turtles were being developed for use in Schools[F]. Since then non wired transmission mediums have become cheaper (and smaller) so are now the normal.

### BBC Buggy

The BBC Buggy launched in 1983[G]. This came in a set that users needed to assemble themselves with a range of programs. This set was modular so kits could be switched out depending on their intended purpose. One such kit was the pen kit that could be used with Logo to draw.

### The Valiant Turtle

Also released in 1983, the Valiant Turtle is controlled using LOGO to draw. It also had an on-screen version[L]. The turtle was controlled with infrared.

### LEGO Mindstorms

In 1987, LEGO started a collaboration with Logo to create turtles for them[B]. The first was the LEGO TC Logo. This required being wired to send commands through its ports. Other ports could be connected to sensors, lights and motors[E].

After this work started to put a processor into a large “brick” so that the bot could store and process its own commands. This led to the LEGO Mindstorms RCX in 1998. This concept was carried forward. Mindstorms are still sold today, though have gone through a few evolutions and now use block-based programming [I].

### TurtleBot

The TurtleBot is more for adult users than the original turtles. They are based upon Logo[J]. People can buy kits to make the robots, and this can lead to diverse applications, including home use and research[K].

## Conclusion

Many of the turtlebots and alternatives have had similar characteristics to the turtlebot for this project. The basic functions are naturally the same as pen controls, and movement requirements have not changed since 1969. The transmission of commands is similar to most other bots mentioned, with the exception of having a different medium, using Bluetooth rather than wired or infrared options. Like the original turtles, the turtlebots produced for the project are also made with the mindset of keeping production costs as cheap as possible.

The turtle graphics library that is used for the project is based upon the original one using the same concepts of position and heading even if the technology has advanced to where it no longer requires a separate graphics terminal.

The aim of turtles has not changed between the originals and this project. They both have the objective of getting beginner programmers to adopt coding by having a go.

1. Reporting on the project – the core chapters

Reporting on the project will normally require more than one chapter.

A development project is likely to have chapters addressing requirements, design, implementation, testing and packaging if a plan-driven method is used. If an agile approach is taking, you might have a chapter for each sprint or iteration.

* Requirements/ changing of them -plan based though flexible to add in new features.
* Spike work
* Design – class diagrams etc.
* Implementation- how (see diary)
* Testing (test table!)
* Packaging

You are likely to include diagrams or images in your core chapters (see Figure 1.)

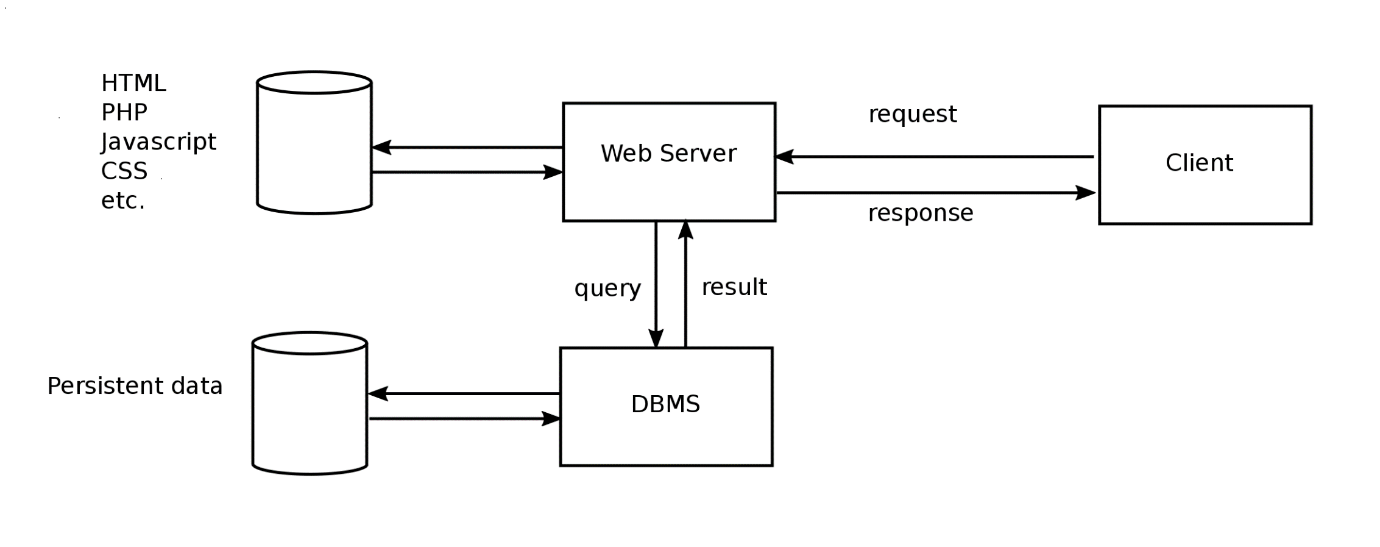


Figure Structure of a dynamic website (Edel Sherratt)

You can refer to your figure from more than one place (Figure 1.)

1. Requirements

## Project requirements

## Methodology

1. Spike work

## Picking software

The two obvious choices for writing the software were using either Java or Python.

### Java

Java had the benefit of already being proved to be able to communicate and send commands to the project.

Java also has a turtle library which can be used to create a virtual turtle and program it with it commands, and with an additional library such as JavaFX, an application could be produced using this.

A screenshot of a computer

AI-generated content may be incorrect.

One major downside of using Turtle with Java is that there is limited documentation for it with no central place with most needed information. This makes it harder to develop a more complicated system with it.

### Python

Python also has a turtle library which can be used in a similar way to the Java one. It also has formal Turtle graphics documentation[x] which can be easily accessed. Spike work was additionally done to check that the turtle could be embedded into a window which could have additional features added to it such as a button. CustomTkinter was used for this given previous familiarity with it.

A screenshot of a computer

AI-generated content may be incorrect.

By creating the application in Python then it allows the users to also write their code in Python. This achieves the application's aim of getting users coding with a proper programming language, whilst making development easier as no translation would be needed between languages.

Python is quite commonly taught in schools at secondary schools so a fair few potential users may already have some familiarity with it. It is popular since it resembles spoken English in many ways so can be easier to pick up. Given this it may prove easier for activities for outreach and open days.

The downside of Python would be that connecting to the turtlebot has not previously been worked out, but given Python’s popularity and large amount of documentation, there will be something out there to achieve this purpose.

### Conclusion

Python was chosen for the reasons listed above. The CustomTkinter library was chosen to build the GUI as a turtle could be easily embedded. CustomTkinter builds upon the Tkinter library which is one of Python’s basic GUI building libraries. CustomTkinter allows Tkinter to be used alongside newer elements such as themes and more modernised GUI features.

## Executing files

For a user to be able to create code and then run it within the application it is crucial to have a way for the code to be retrieved and run dynamically. Python has multiple libraries and ways to do this with their own benefits and drawbacks.

These were explored for their suitability by having a simple python file that would print hello world if executed correctly. Another file with the chosen method to read in the file and execute it was then created. These were tried both with the hello world file being correct and when it has errors.

Once the options had been explored this way, they were tried with a CustomTkinter window containing a textbox were “print(“Hello world”)” was written and a button that called a command to run the code that was written. The command was changed to use each library option.

A screenshot of a computer error

AI-generated content may be incorrect.

### The subprocess module

The subprocess library has multiple functions that were tested out. Call, run and Popen were all tested out. These all did not output the expected result of “hello world”. By using the check\_output function and then printing out its results, it can be found to return "b'hello world\r\n'”. After investigation, the subprocess library seems to be more for running command line processes rather than Python files making it not suitable for this project.

### The os module

The os module offers the system function that can be used to read and execute files. It proved to be effective for the first text, printing “Hello world” to the console. Once CustomTkinter was introduced it stopped functioning.

### The exec function

The exec function is built into Python’s basic functions and executes the string of code it is given. To use this the python file must be opened and read then the contents passed to the exec function. All these functions are within Python’s standard library. This method worked successfully for both tests. For the first test, code errors in the hello world file caused the system to stop. This was not a problem once CustomTkinter was introduced as the application would keep running, although throwing errors when the code was wrong.

This method has the original benefit that it does not require the input to come from a file, as the exec function is looking for a string. This means that the code can be retrieved straight from the textbox and passed to it, skipping the middle steps.

The main issue with the exec function is that it will try to run any code it is given. This means it has security issues though since the use of the application is likely to be monitored it should be a minimal issue for this application.

The exec function shall be used as it works for desired application.

## Threading

Research was done into threading with Python, as the application would require multiple processes to be handled at once. Python is single threaded by default, and many libraries rely on being on the main thread, such as CustomTkinter. Self-made processes can be run on separate threads. This was attempted, utilising the turtle module to get two turtles to run in parallel to test using the threading library. This was successful, so it will be usable for more complicated parts of the final system.

A screen shot of a computer screen

AI-generated content may be incorrect.

## Syntax highlighting

For a better experience, coding it was decided that syntax highlighting would be useful for the box where users enter their code. This could be done for known keywords, such as for or strings, but would be better done with a library, as all the main keywords and syntax would already be known. There are a few libraries that serve this purpose, two of the most common ones for use with Tkinter are Idlelib and Pygments. Both libraries worked for their purpose when tried out, with the same result from the user's point of view.

Given that the application is intended to be kept simple and easily maintainable for developers, Pygments proved bulkier and required more lines of code to achieve the same purpose. Idlelib is more streamlined from a developer using its point of view and allows for easy choosing of colours for each kind of syntax highlighting that is wanted.

## HTML pages

The user will require some guidance about what code they can write and may wish to learn more about the software and hardware. To this end, some About pages would prove beneficial. To do this, it was decided to write the information in HTML for the application to then open and display on a separate window. This way, the pages could be updated without prior Python knowledge and without having to open the main code. A library that could provide this function was therefore required.

There are two main libraries that can open HTML pages inside Tkinter windows, both of which have their own benefits and drawbacks.

### tkhtmlview

The tkhtmlview library is specifically designed to work with Tkinter. HTML pages can be opened using it by putting files into objects such as its HTMLScrolledText, which can then be packed and displayed in a window. The main drawback of this library is that it doesn’t support the navigation tag. When used, it tries to open an unreachable webpage.

### Webview

Webview works by creating a new window using the HTML file. This works when it stands alone, and the navigation tag can be used. Problems arise when attempting to move between the About pages and the main window. Doing that causes the application to crash. Since the pages are designed to be referred to for command options whilst typing them, this is less than ideal. The Webview library only functions on the main thread, so putting it onto an alternative thread was not an option.

### Conclusion

The tkhtmlview library was chosen to be used since it mostly worked, and its drawback was minor in comparison.

1. Design

## Use cases

## UI discussion

## Classes

## Files

## Setup wizard

The setup wizard is designed so that each turtlebot can be configured to the correct dimensions. These dimensions can vary slightly mainly due to shrinking as the plastic cools down once the parts are 3D printed. When the dimensions are slightly off it can lead to results such as circles and squares not matching up or lines drawn not being the expected length. Although this is minor was smaller drawings, when larger drawings are attempted this error is significant.

To stop these problems a setup wizard is produced so that a user use it to work through various steps to calibrate the turtlebot to the correct dimensions and values. The dimensions in question are both wheels diameters and the distance between the two wheels (axle length). It is also important to measure the backlash of the motors so they can be factored into any lines to be drawn.

The setup wizard works in three stages: working out the backlash, working out the wheels’ diameter and working out the axle length (see Figure 2). After the dimensions are calibrated to a sufficient level they are saved to the EEPROM. After this, the turtlebot loads these values to use within the main program upon future startups.

A diagram of a process

AI-generated content may be incorrect.

Figure - Setup wizard flow chart

### Backlash

The first calibrations are done for the turtlebot’s backlash. This is done by having the turtlebot move backwards and then try to move it forward. If the turtlebot moves then it was instructed to move more than its backlash.

A diagram of a flowchart

AI-generated content may be incorrect.

Figure - Setup wizard flow chart: Setting the backlash

As seen in Figure 3, the turtle originally moves back by a large amount of steps. The increment starts as a fifth of the max amount. The turtle attempts to move forward by this amount.

If no forward movement occurs, then the turtle attempts to move forward by another increment amount. If movement does occur and a fifth of the increment is a whole number then the increment becomes a fifth of itself. The turtle then moves backwards by the last value it moved forward by and the moving forward process starts again.

If the increment becomes less than one, then the new backlash has been found and is saved. This is because the backlash is saved as the number of motor steps, which can only be a whole number.

### Wheel diameter

The wheel diameter can be worked out whether a line drawn is the correct length. If a straight line drawn is shorter than expected then the wheel diameters are smaller than expected. A longer line indicates that the diameters are bigger than expected. We assume that the diameters of both wheels are equal. This is likely as they are printed at the same time so should have been effected equally by factors. If this is not true, it also becomes apparent during the line drawing test, though it cannot be fixed using the setup wizard.

A diagram of a flowchart

AI-generated content may be incorrect.

Figure - Setup wizard flow chart: Wheel diameters

As seen in Figure 4, the user prompts the turtle to draw a line that should then measure 300mm. The user then uses a ruler to measure the actual length of the line and inputs this value. If this value matches the expected value, then the diameters are already correct, so the wizard can move to the next step. If this value does not match, then the new diameters are calculated by the fraction that the length was off by multiplied by the expected diameter. This new diameter is saved and the user can have the turtle draw the line again to see if the new diameter value is correct.

### Axle Length

Axle length becomes important when the turtlebot attempts turns. If the axle length is shorter than expected then the turtlebot will under turn and vice versa. This leads to shapes not connecting properly, either forming a gap, overlap or cross shape.

A diagram of a flowchart

AI-generated content may be incorrect.

Figure - Setup wizard flow chart: Axle length

As seen in Figure 5, a sequence of steps is undertaken to calibrate this. The turtlebot first draws two circles. Each circle uses one wheel as a pivot point while moving the other wheel to draw a circle. Each circle is calculated by working out how many wheel motor steps need to be taken to do a circle with the diameter of the axle. This is done by dividing the circumference of an axle diameter circle by the circumference of a wheel diameter circle and then multiplying this by the number of steps in a full wheel rotation. An example of the circles drawn can be seen in Figure \_\_\_\_.

The user is then prompted to input two length values. If there is a gap in the circle the user inputs how much by as a negative. If there is an overlap the user puts in how much by as a positive number. If the circles are good then the user can input two zeros, in which case the setup wizard will move onto finishing. If there is a gap or overlap, the two inputted values are averaged. The expected value then has this value divided by 2(?) pi subtracted from it to get the new axle value. The user can then repeat the drawing process in hopes of seeing an improvement.

1. Implementation
2. Testing

## How testing was approached

* Manual testing cause UI
* Unit tests not appropriate enough to use, functions rarely return anything an when they do it’s a UI element.

## Conclusions

1. Packaging?
2. Critical Evaluation

The critical evaluation consists of a discussion, leading to conclusion. It is an essential part of a master’s degree.

It shows that you can not only carry out a substantial piece of work, but that you can reflect on it, and think critically about how you might have done it better.

Examiners view the critical evaluation as very important.

Critical evaluation should contain.

* Strengths and weaknesses of your project
* If you were unable to attain any deliverables, then why.
* What are the future plans for your project if you are to continue

You will be presenting this during demonstration but here you need to discuss them in detail.

1. Conclusion

A brief summary of all that has gone before, including the key results of the project.

May also include some directions for future work.

# References

|  |  |
| --- | --- |
| [1] | I. Jones, “New kinds of red ink,” *Inky Journal of Pigments,* vol. 336, no. 5, pp. 55-58, March 2010. |
| [2] | M. Copilot, *Prompt: What is green IT, https://www.bing.com/chat,* Accessed: 24 February 2025, 2024. |

# References

[A] S. Papert, “A COMPUTER LABORATORY FOR ELEMENTARY SCHOOLS,” MASSACHUSETTS INSTITUTE OF TECHNOLOGY A.I. LABORATORY, Oct. 1971. Accessed: Jul. 30, 2025. [Online]. Available: <http://hdl.handle.net/1721.1/5834>

[B] C. Solomon *et al.*, “History of Logo,” *Proceedings of the ACM on Programming Languages*, vol. 4, no. HOPL, pp. 1–66, Jun. 2020, doi: <https://doi.org/10.1145/3386329>.

[C] C. Solomon, “Logo Things,” logothings, 2016. https://logothings.github.io/logothings/Home.html#HOME (accessed Aug. 06, 2025).

[D] P. Boytchev, “About,” Uni-sofia.bg, 2015. https://pavel.it.fmi.uni-sofia.bg/logotree/about.html (accessed Aug. 06, 2025).

[E] “LEGO® TC logo.” Accessed: Aug. 06, 2025. [Online]. Available: https://ia802903.us.archive.org/4/items/lego-tc-logo-reference-guide/lego-tc-logo-reference-guide.pdf

[F] S. Papert and C. Solomon, “TWENTY THINGS TO DO WITH A COMPUTER,” pp. 1–5, Jun. 1971. Accessed: Aug. 06, 2025. [Online]. Available: https://dspace.mit.edu/bitstream/handle/1721.1/5836/AIM-248.pdf?sequence=2&isAllowed=y

[G] N. Fazakerley, “BeebControl’s BBC Buggy Page,” Riscy.uk, 2025. http://www.riscy.uk/beebcontrol/buggies/bbc\_buggy/index.html (accessed Aug. 06, 2025).

[H] Scratch, “Scratch - Imagine, Program, Share,” *Scratch*, 2007. <https://scratch.mit.edu/> (accessed Aug. 07, 2025).

[I] “LEGO® MINDSTORMS® About | Official LEGO® Shop GB,” *www.lego.com*. <https://www.lego.com/en-gb/themes/mindstorms/about> (accessed Aug. 07, 2025).

[J] Evan Ackerman, “TurtleBot Inventors Tell Us Everything About the Robot,” *IEEE Spectrum*, Mar. 26, 2013. https://spectrum.ieee.org/interview-turtlebot-inventors-tell-us-everything-about-the-robot

[K] TurtleBot, “TurtleBot,” *www.turtlebot.com*. <https://www.turtlebot.com/> (accessed Aug. 07, 2025).

[L] The Centre for Computing History “Valiant Turtle - Peripheral - Computing History,” *Computinghistory.org.uk*, 2025. https://www.computinghistory.org.uk/det/39597/Valiant-Turtle/ (accessed Aug. 07, 2025).

‌

‌

Appendix A

Generative AI

1. No AI was used for the project.

Third Party Code and Software Libraries

1. .NET’s ASP.NET libraries have been used for this project. The document template created by the dotnet CLI tool produces a set structure for the Model-View-Controller parts of the project. The CLI tools were also used to generate the Entity Framework Core code in most Controllers. The CLI generated code was then adjusted to make it relevant to this application.

Appendix B- Manual testing table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Ref | Test Content | Input | Output | Pass Criteria | Pass/ Fail |
| T-01 | Does the application start | Click on the Turtlebot.exe | The application opens with the default values and layout as shown: | No port is currently displayed in the top right, along with two red boxes. The default code of “turtle.down()  turtle.forward(20)  turtle.right(90)  turtle.up()  turtle.forward(20)” is in the textbox |  |
| T-02 | Can the default code be run on the simulation? | Click on run after launching the application | The right-hand side shows the turtle drawing and the box underneath the run button displays the commands in the input screen as they are carried out |  |  |
| T-03 | Can the virtual turtle be reset? | Click on the Reset button in the turtle box’s top right corner after state left by T-02 | The turtle returns to its default position, and the output text disappears | No more lines are visible |  |
| T-04 | Does a dialogue appear when trying to clear the program? | After the state left by T-02, click the “Clear Program” button in the top right of the code box. | A confirmation dialogue is shown asking “Are you sure you want to clear the program?” | The user is presented with two buttons on the dialogue, one for Yes and one for No |  |
| T-05 | Can clearing the program be done? | After T-04, click Yes | The dialogue box disappears, and the code box is now empty | The turtle is reset, and no drawings can be seen. The output box is also blank |  |
| T-06 | Can clearing the program be cancelled? | After T-04, click No | The dialogue box disappears. | No other change occurs |  |
| T-07 | Can a pop-up be opened to select a port? - no ports available | After launching the application:   1. Click settings. 2. Click Select port when no USB ports are available | A dropdown is shown with “No ports” and a Refresh ports button | Shows until Refresh ports is pressed |  |
| T-08 | Can refresh be pressed to see new available ports? | After T-07:   1. Insert the USB dongle. 2. Click Refresh ports | Select port is now shown on dropdown | Ports are available if click on dropdown |  |
| T-09 | Can a pop up be opened to select a port? - ports available | After launching the application:   1. Click settings. 2. Click Select port when USB ports are available | A pop-up containing dropdown is shown with “Select port” and a Refresh ports button | Ports are available if click on dropdown |  |
| T-10 | Can a port be selected? | After the state of either T-08 or 09:   1. Click the dropdown. 2. Click on COM3 | The popup disappears, COM3 is displayed in the white box on the top bar | The red box next to the word USB in the top bar turns green |  |
| T-11 | Can the application show that the USB connection has been stopped? | After T-10, unplug the USB dongle | The green box next to the word USB in the top bar turns red | The box next to Turtle also turns red |  |
| T-12 | Can the turtlebot be connected? | After T-10, press the button on the turtlebot on the right of the display | The box next to the word Turtle turns green from red. | The turtlebot displays the word “Hello” |  |
| T-13 | Can the turtle move the pen down? | Following the state after T-12:   1. Type “turtle. down()” in the code box 2. Press run. | The turtlebot moves its pen down and, “turtle.down()” is displayed in the output box. | No visual differences to the virtual turtle. |  |
| T-14 | Can the turtle move forward? | Following the state after T-13:   1. Type “turtle. forward(20)” 2. Press run | The virtual turtle moves forward and “turtle.forward(20)” is displayed in the output box. | The turtlebot moves forward 20mm. |  |
| T-15 | Can the turtle move forward? | Following the state after T-13:   1. Type “turtle. forward(55)”   Press run | The virtual turtle moves forward and “turtle.forward(55)” is displayed in the output box. | The turtlebot moves forward 55mm. |  |
| T-16 | Are spelling errors for the turtle caught? | Following the state after T-13:   1. Type “turte. forward(20)” 2. Press run. | The output box displays “name 'turte' is not defined” | Nothing else happens |  |
| T-17 | Are spelling errors for methods caught? | Following the state after T-13:   1. Type “turtle. foward(20)” 2. Press run | The output box displays “'User\_Turtle' object has no attribute 'foward'” | Nothing else happens |  |
| T-18 | Are syntax errors caught? | Following the state after T-13:   1. Type “turtle. forward(20” 2. Press run | The output box displays “'(' was never closed (<string>, line 2)” | Nothing else happens |  |
| T-19 | Can the turtle move backwards? | Following the state after T-13:   1. Type “turtle. forward(-20)” 2. Press run | The virtual turtle moves backwards, “turtle.forward(-20)” is displayed in the output box. | The turtlebot moves backwards 20mm. |  |
| T-20 | Can the turtle move the pen up? | Following the state after T-12:   1. Type “turtle.up()” 2. Press run | The turtlebot moves its pen up and, “turtle.up()” is displayed in the output box. | No visual differences to the virtual turtle. |  |
| T-21 | Can the turtle move 90° right? | Following the state after T-12:   1. Type “turtle.right(90)” 2. Press run | The turtlebot turns 90° to the right and, “turtle.right(90)” is displayed in the output box. | The virtual turtle also turns right so it is now facing south. |  |
| T-22 | Can the turtle move 180° right? | Following the state after T-12:   1. Type “turtle.right(180)” 2. Press run | The turtlebot turns 180° to the right and, “turtle.right(180)” is displayed in the output box. | The virtual turtle also turns right so it is now facing West. |  |
| T-23 | Can the turtle move 360° right? | Following the state after T-12:   1. Type “turtle.right(360)” 2. Press run | The turtlebot turns 360° to the right and, “turtle.right(360)” is displayed in the output box. | The virtual turtle also turns right so it is now facing East. |  |
| T-24 | Can a negative value be entered for moving right? | Following the state after T-12:   1. Type “turtle.right(-90)” 2. Press run | The turtlebot turns 90° to the left and, “turtle.right(-90)” is displayed in the output box. | The virtual turtle also turns left so it is now facing North. |  |
| T-25 | Can the turtle move 90° left? | Following the state after T-12:   1. Type “turtle.left(90)” 2. Press run | The turtlebot turns 90° to the left and “turtle.left(90)” is displayed in the output box. | The virtual turtle also turns left so it is now facing north. |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | Curve | (0,0) |  |  |  |
|  |  | (0,360) |  |  |  |
|  |  | (360,0) |  |  |  |
|  |  | (100,60) |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| T-22 | Can a variable be used for a forward value? | Following the state after T-12:   1. Type “x=50” 2. Add a new line. 3. Type “turtle.forward(x)” 4. Press run | The virtual turtle moves forward, “turtle.forward(50)” is displayed in the output box. | The turtlebot moves forward 50mm. |  |
| T-23 | Can a variable be used for a right value? | Following the state after T-12:   1. Type “x=180” 2. Add a new line. 3. Type “turtle.right(x)” 4. Press run | The turtlebot turns 180° to the right and, “turtle.right(180)” is displayed in the output box. | The virtual turtle also turns right so it is now facing west. |  |
| T-24 | Can a variable be used for a left value? | Following the state after T-12:   1. Type “x=180” 2. Add a new line. 3. Type “turtle.left(x)” 4. Press run | The turtlebot turns 180° to the left and, “turtle.left(180)” is displayed in the output box. | The virtual turtle also turns left so it is now facing west. |  |
| T-X | Can for loops be used? | Following the state after T-12:   1. Type “for i in range(4):” 2. Add a new line. 3. Add a tab. 4. Type “turtle.forward(20)” 5. Add a new line. 6. Add a tab. 7. Type “turtle.right(90)” 8. Press run | The turtlebot draws a square of dimensions of 20mm by 20mm. The output box displays “turtle.down()  turtle.forward(20)  turtle.right(90)  turtle.forward(20)  turtle.right(90)  turtle.forward(20)  turtle.right(90)  turtle.forward(20)  turtle.right(90)” | The virtual turtle draws a square: |  |
| T-26 | Are indent errors caught? | Following the state after T-12:   1. Type “for i in range(4):” 2. Add a new line. 3. Type “turtle.forward(20)” 4. Add a new line. 5. Type “turtle.right(90)” 6. Press run | The output box displays “expected an indented block after 'for' statement on line 2 (<string>, line 3)” | Nothing else happens |  |
| T-27 | Can the turtle be stopped? | Straight after T-X press stop | The turtlebot stops after the current command. The output box says stop safter the current command. | The virtual turtle stops after the current command. Nothing else is added to the output box. The run button is available to click again. |  |
| T-Y | Can a save as dialog be opened | Following the state after T-X:   1. Click on File on the top bar 2. Click Save |  |  |  |
| T-29 | Can the current code be saved? | Following the state after T-Y:   1. Enter “my\_code” as the filename 2. Click Save |  |  |  |
| T-30 | Can saving the file be cancelled | Following the state after T-Y:   1. Click Cancel |  |  |  |
| T-31 | Can code be loaded? |  |  |  |  |
| T-32 | Can loading code from a file be cancelled? |  |  |  |  |
| T-33 |  |  |  |  |  |
| T-34 |  |  |  |  |  |
| T-36 |  |  |  |  |  |
| T-37 | Insert number |  |  |  |  |
| T-38 | Change font size |  |  |  |  |
| T-39 |  |  |  |  |  |
| T-40 |  |  |  |  |  |
| T-43 | Pen height |  |  |  |  |
| T-44 | Pen height up |  |  |  |  |
| T-45 | Pen height- down |  |  |  |  |
| T-46 | Setup wizard- no port |  |  |  |  |
| T-47 | Setup wizard port |  |  |  |  |
| T-48 | Setup wizard- backlash |  |  |  |  |
| T-49 | Setup wizard- backlash |  |  |  |  |
| T-50 | Setup wizard- backlash |  |  |  |  |
| T-51 | Setup wizard- diameter |  |  |  |  |
| T-52 | Setup wizard- short |  |  |  |  |
| T-53 | Setup wizard- long |  |  |  |  |
| T-54 | Setup wizard- correct |  |  |  |  |
| T-55 | Setup wizard- diameter |  |  |  |  |
| T-56 | Setup wizard- short |  |  |  |  |
| T-57 | Setup wizard- long |  |  |  |  |
| T-58 | Setup wizard- correct |  |  |  |  |
| T-59 | About | After T-01, click on the About button on the top bar | A window opens with additional information which matches the layout defined in the HTML file. | The main window can be used (clicked on and typed on) whilst the About window is open |  |