

Integrating Project Report

by Jamie Oliver

Submission date: 03-May-2024 02:03PM (UTC+0100)

Submission ID: 232095487

File name: 41399_Jamie_Oliver_Integrating_Project_Report_573445_525177054.docx (744.86K)

Word count: 2190

Character count: 10504

Lego Spike Prime Project

L4: Integrating Project – Computing

Module Code: ACHA4008

BSC4F Computing

(Games Development)

Contents

1. Introduction	3
2. Researching Tools & Kit.....	3
i. Building Kit.....	3
ii. Word Blocks.....	3
3. Researching Navigation Methods.....	4
i. Following Lines	4
ii. Navigating Obstacles	4
iii. Escaping a Maze.....	4
4. Planning	5
5. Implementing and Testing	5
i. Building the Robot.....	5
ii. Line Following	6
iii. Maze Solving.....	7
References	8

1. Introduction

In this project, our objective was to create a robot using the Lego Spike Prime kit, and program it to navigate certain paths, obstacles, and finally, a maze. In order to do this, the class was split into groups, each of which worked together to familiarise ourselves with the Lego Spike Prime building kit, and learn its associated code blocks which are used for programming it. This report will document this process, and how it lead to our final bot's design.



2. Researching Tools & Kit

This project was facilitated via the use of the Lego Spike Prime kit, released in August 2019 [1] by Lego. The kit was designed to be used in classrooms to inspire and encourage kids to engage with and pursue Science, Technology, Engineering, Artistic, and Mathematical based subjects and careers. For our current project, we had to physically design a robot with the Lego blocks, as well as program it to be able to navigate and solve problems.

i. Building Kit

Each building kit provided with Spike Prime includes 528 elements [2] to use, including a hub and various kinds of motors and sensors. The kit came with designated storage for all the various kinds of bricks and pieces, as well as an illustrated reference sheet  to keep track of them all. Over several weeks in class, we were able to familiarise ourselves with these parts by building and testing various models, such as driving bases, and grabbing mechanisms. Becoming familiar with the parts and getting practical experience using them meant that we could apply what we had learned to new problems, and equipped us with the knowledge of how to physically build and modify these Lego constructions, and how their sensors and motors could be utilised.

ii. Word Blocks

In order for the Lego Spike Prime models to actually function, we needed to program its various mechanisms. The class spent several weeks trying out Lego's own models and instructions for things like the driving base and grabber. This allowed us a greater understanding of these word blocks, which are Spike Prime's own native code language, and allowed us to be creative and versatile with what we programmed our bot to do. This meant finding and using blocks that were related to a specific sensor or motor, and applying the right functions depending on the intended effect. For example, if we wanted the bot to drive until it sensed red, we would have to enable its movement motors, and then use different word blocks upon sensing red with the colour detector.



[3]

James Oliver (2212598)
2212598@student.uwtsd.ac.uk

3. Researching Navigation Methods

When developing our plan for tackling this maze-navigating robot, we divided the tasks between ourselves. While the others in our group were in charge of researching the methods to be used to navigate the line, and building the robot, I lead the task of translating these methods into code and programming maze navigation of the robot.

i. Following Lines

The first part of our task was finding a method for the robot to follow a line on the ground from one point to another. Since the robot also needed to know when to stop or start depending on a coloured sheet on the floor, we chose to use the colour sensor to tackle this. This meant fixating a colour sensor to the bottom of the robot, so it could differentiate between line, floor, and coloured paper.

ii. Navigating Obstacles

The next part of the task involved making sure our robot had a method to navigate around a block in its path. We had previously worked on a grabbing mechanism in class as practice [3], using the pressure sensor on the robot. This seemed like the best approach to this issue, as the sensor could work as a button to detect if the bot has bumped into an obstacle. To facilitate this, we decided the pressure sensor would be fixed onto the front of the bot.

iii. Escaping a Maze

The final method for the robot was for it to navigate and escape a maze. To begin with, we researched the most effective way to find the exit of a maze, and discovered that the 'hand-on-wall' [4] was the most commonly used method. This method entails placing either the left or right hand on the wall and following it without letting go. Doing ensures that eventually, the user will either find an exit or end up back at the beginning, as every wall of the maze must link back to an exit, and keeping your hand on the wall prevents you taking any unnecessary detours or getting lost by disconnecting from the wall you came in following. To facilitate this on our robot, we decided to add a proximity to the left side of it. This meant it could technically follow the left-hand rule, by turning left whenever there is a space to do so [5], and only turning right if it bumps into a wall with the pressure sensor on the front.



[4]

4. Planning

To plan out our time for this project, we divided it into three main segments; building the robot, coding the line-following, and coding the maze navigation. Since the latter two were more complex, we decided to split each of them into three more parts each; writing the initial code, debugging and testing the code, and evaluating the outcome once it was working.

We allocated time to each of these sections to ensure they were all worked on to some degree without forgetting to do another. To begin with, we set an hour for building the robot itself, as this would be a relatively simple task. Then we cycled between the line and maze sections of the robot so that each would be worked on equally for the time leading up to the demonstration. This meant writing the initial code for the line first, and then the maze, and then debugging and testing both of them one after the other, switching often so that any problems we overcame or solutions we learned could be applied to the other if needed. We then decided after each was finished and working, we would evaluate it and make note of any changes for the future.

5. Implementing and Testing

Now that we knew our methods of navigation, we needed to implement them practically by building the robot and writing its code.

i. Building the Robot

To give ourselves a good base for our robot, we used Lego's driving base [6] as a reference for the physical frame. We fixed motors to the left and right of the hub and attached wheels to these, and fixed a gyro to the front for turning. We made sure the wheels and gyro were low enough to accommodate the colour sensor, which was also fixed to the bottom so it could sense the lines. We then fixed the pressure sensor to the front to sense collisions, and added the proximity sensor to the top facing left, so the bot could check if a wall was near it or not.

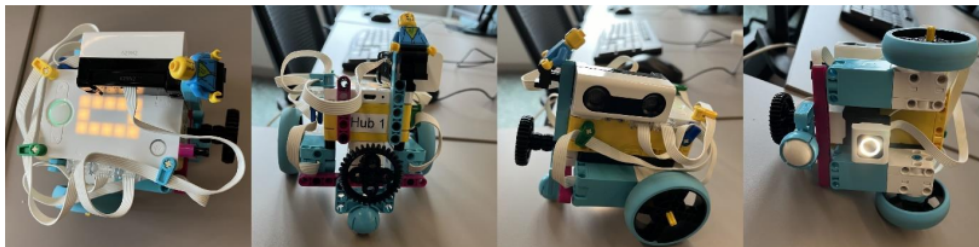


Figure 1 - Robot Built. 2, 3, 4 show Pressure Sensor, Proximity Sensor, and Colour Sensor respectively.

ii. Line Following

To code the line-following section of the robot, I defined three separate code blocks so each part of the program was distinct and easier to debug.

The simplest part was defining the conditions for this section to stop. Since the robot would enter the maze by going over a green piece of paper, we simply added a code block that exits and ends all line-following code once the colour sensor detects green.

The next code block gave the robot a way to navigate around a block in its path. This code block was triggered when the pressure sensor was pressed, indicating a collision. The code then pauses any other movement and begins to navigate around the obstacle. It does this by simply turning 90 degrees to the right and moving forward to the side of the block, turning 90 degrees again and moving along the side of the block, and then 90 degrees to the left again, and moving forward to join the line again, and turning to face the direction it was originally. This gives the robot a range to get around the block, while ensuring it ends up joining the line ready to move forward.

The final code block was for regular movement following a red line. This was performed by checking whether the colour sensor could detect red, and moving forward if it can. If the colour sensor did not detect red, it would turn right for 90 degrees, or until it detect red. If it did not detect red, it would perform the same action while turning to its original position, and then turning left. This gives the robot a 180 degree scope while it tries to find the line again if it has lost it.

All of the above code blocks also had lights associated with their action, so we would know what action was being performed by the hub as it moved. For example, right turns displayed an "R", whereas left turns displayed "L". This made debugging the bot a lot easier as we worked with it, because if a certain part wasn't working we could review which light was displayed and view the code for it.

We tested this code by running our robot over a red line to determine its path. After some trial and error with the code, we found it was successfully able to navigate this red line. However, there were some flaws in our design. Namely, since the robot always turned right before left when checking for the line, it would have to rotate multiple times before every movement if turning left or following a curve. Next time to remedy this, we could have the robot randomly determine a direction to turn, or choose based on the direction of the last successful turn.

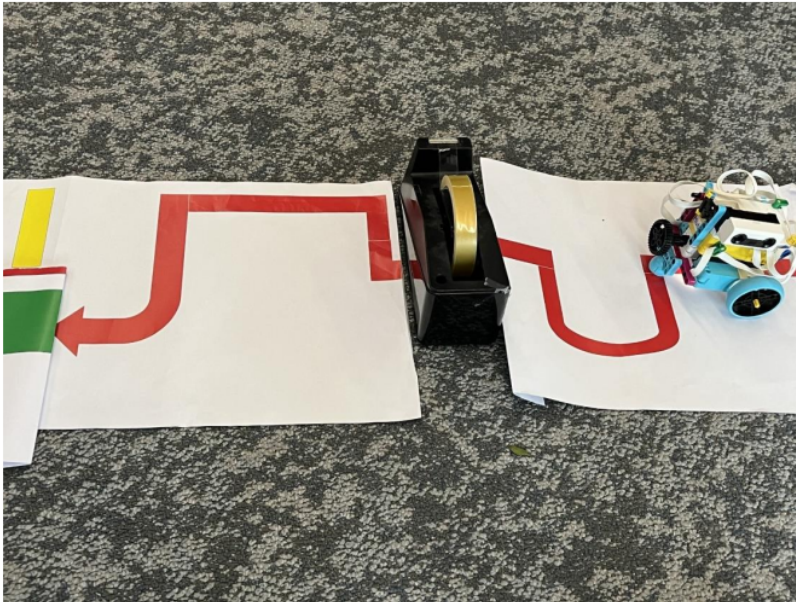


Figure 2 - Robot navigating a red line & an obstacle

iii. Maze Solving

Coding the maze was similar to the line, and I split the processes into three code blocks for each type of action it would perform.

The first code block was simply moving forward. In this, the bot s  5 checks whether any of the other code blocks for turning are active, and moves forward if they are not. The colour sensor also checked for green, to avoid exiting the maze, and red so it knows when it has found the exit.

The second code block is for turning right. This block is triggered when the bot hits a wall and triggers the pressure sensor. It then moves back from the wall slightly, and turns 90 degrees to the right before continuing.

The third code block is for turning left. This was triggered when the distance sensor was further than 20cm from a wall. This allowed the bot to turn 90 degrees left into any gaps it detected, before continuing.

Each of these segments had lights associated with their action, similar to the line following code, so we could debug any issues.

We tested this code by placing our robot in a makeshift plywood maze. Through some trial and error, we found this model was successfully able to navigate its way out of the maze. The bot successfully followed the 'hand-on-wall' technique and could cope with variations of the maze layout to make its way to the exit. We did have some issues, notably with the bot struggling to make precise turns, as it relied on entering the maze with a yaw equivalent to a multiple of 90, otherwise its turns were slightly skewed and offset in the maze and it would bump into walls often. To remedy this, we could pull the wheels in tighter to the body to avoid collisions when

[7]

James Oliver (2212598)
2212598@student.uwtsd.ac.uk

turning around walls, and refine the code so that it can navigate the maze easier even if it didn't enter perfectly straight.

Figure 1 - Robot Built. 2, 3, 4 show Pressure Sensor, Proximity Sensor, and Colour Sensor respectively. 5

Figure 2 - Robot navigating a red line & an obstacle..... 7

References

[1] Graham, "Lego Education unveils Spike Prime," Brick Fanatics, 2 April 2019. [Online]. Available: <https://www.brickfanatics.com/lego-education-unveils-spike-prime/>. [Accessed 29 April 2024].

[2] "Lego Education Spike Prime Set," Lego, [Online]. Available: <https://education.lego.com/en-gb/products/lego-education-spike-prime-set/45678/>. [Accessed 29 April 2024].

[3] "Grabbers," Lego, [Online]. Available: <https://spike.legoeducation.com/prime/models/bltbcaaaa2cd334d8a9>. [Accessed 30 April 2024].

[4] E. Roberts, "Recursive Backtracking," 23 January 2015. [Online]. Available: <https://cs.stanford.edu/people/eroberts/courses/cs106b/handouts/16-RecursiveBacktracking.pdf>. [Accessed 30 April 2024].

[5] "Left hand on the Wall," Pololu Robotics & Engineering, [Online]. Available: <https://www.pololu.com/docs/0J21/8.c>. [Accessed 30 April 2024].

[6] "Driving Base 1," Lego, [Online]. Available: <https://spike.legoeducation.com/prime/models/bltc58e302d70cf6530>. [Accessed 30 April 2024].

Integrating Project Report

GRADEMARK REPORT

FINAL GRADE

71 /100

GENERAL COMMENTS

Your project report showcases a your basic understanding of robotics however there is room to deepen your analysis.

Here is some feedback for your report:

Positives:

- Your plan is structured well, making it easy to track your progress.
- each phase of your project clearly assessed.

Suggestions for Improvement:

- Providing more details about the specific challenges you encountered and how you addressed them would be beneficial.

Keep up the good work and continue delving into the fascinating world of robotics!

Introduction- 11

Research skills- 9

Plan-34

PAGE 1

PAGE 2

PAGE 3



Comment 1

well analysed however more specifics about the assessment could have been added



Comment 2

more details about the sensors and the components could have added credits



Comment 3

In professional or formal writing, it is more appropriate to use the term "robot" instead

PAGE 4



Comment 4

well analysed and explained

PAGE 5

PAGE 6

PAGE 7



Comment 5

code blocks could also have been added

PAGE 8
