

Integration Project

by Alin Astefanei

Submission date: 05-Apr-2025 01:05PM (UTC+0100)

Submission ID: 254892054

File name: 58288_Alin_Astefanei_Integration_Project_660190_615969334.docx (1.29M)

Word count: 2054

Character count: 11913

**Integrating Project (Computing Foundation Year
version)**

ACHA4008

Report Writing

Content

1. Introduction

An overview of the project
(Page 2)

2. Background Research

Research into line-following techniques, maze-solving methods,
and sensor usage.
(Page 2-3)

3. Project Plan

A breakdown of the steps taken
(Page 4-5-6)

**4. Design Process- (Robot Design, Code design, maze
navigation**

Explains how the robot's hardware and software were designed
(Page 6-7-8-9-10-11)

5. Evaluation and Results

(Page 11-12-13-14-15)
Reviews how well the robot performed, what **worked well**, what
could have been **improved**, and ideas for **future enhancements**.

Student Alin Ilie Astefanei (2404824)

1. Introduction

Autonomous navigation is a key challenge in robotics, used in applications like self-driving cars and warehouse automation. This project involves designing and programming a **LEGO SPIKE robot** to follow a line and navigate a maze using sensors and control algorithms.

The robot is built with **motors, colour sensors, and the SPIKE Prime Hub**, with programming done in **Word Block or Python**. This report outlines the **research, design, implementation, and evaluation** of the project, examining key techniques for line-following and maze-solving.

2. Background research (Theories, Techniques and Tools)

• Line-Following Techniques

Colour Sensors & Contrast Detection:

- The LEGO SPIKE robot uses colour sensors to detect the contrast between a black line and a white surface.
- The sensor reads light intensity values to determine whether the robot is on the path.

• Basic Line-Following Algorithm:

- The robot continuously adjusts its movement based on sensor readings.

García, M., et al. (2020). "Line-Following Robot Performance Using Colour Sensors: A Comparative Study." *IEEE Robotics & Automation Magazine*, 27(4), 78-89.

- PID (Proportional-Integral-Derivative) Control: *(Optional, if using advanced techniques)* **(Nguyen, T., et al. (2021).** "Optimizing PID Control for Line-Following Robots in Competition Environments." *IEEE Transactions on Control Systems*, 68(5), 320-334.)

- Helps smooth movement by reducing oscillations and improving accuracy.

• Maze-Solving Techniques

Wall-Following Method (Right-Hand or Left-Hand Rule):

-The robot follows one wall (left or right) to find its way through the maze.

Dead-End Detection:

-If the robot reaches a dead end, it needs to **turn around** and find an alternative path.

Decision-Making at Intersections:

-The robot detects junctions and chooses the correct turn using pre-programmed logic.

Wilson, J. (2019). "Maze-Solving Methods in Robotics." Robotics & AI Journal.

LEGO SPIKE Components Used

-**Motors:** Control the movement of the wheels.

-**Colour Sensors:** Detect the line for navigation.

-**SPIKE Prime Hub:** The brain of the robot, processing sensor data and executing commands.

Programming Choices: Word Block vs. Python

- **Word Block (Block-Based Coding):**

-Easier for beginners, uses drag-and-drop blocks.

- **Python (Text-Based Coding):**

-More flexible, allows for advanced logic like PID control.

3. Project plan

Steps to complete the project

The project is divided into several key phases:

1st Research & Planning (Week 1-2)

- Study line-following and maze-solving techniques.
- Select programming language (Word Block or Python).
- Plan robot design and sensor placement.

2nd Hardware Setup (Week 2-3)

- Assemble the LEGO SPIKE robot.
- Install and position colour sensors.
- Test motor movements (forward, turn, stop).

3rd Software Development (Week 3-4)

- Program basic line-following logic.
- Implement maze-solving algorithm (wall-following or decision-making).
- Optimize movement (e.g., speed adjustments, smoother turns).



4th Testing & Debugging (Week 3-4)

- Run multiple trials to check performance.
- Identify and fix sensor inaccuracies or navigation errors.

5th Final Evaluation & Report Writing (Week 4-5)

- Analyse test results (success rate, errors).
- Suggest improvements.
- Compile findings into the final report.

Task allocation

The task has been divided in 3 parts which has been cover by the whole team created by 3 Students (Alin, Ahmad and Hossam)

Research Leader (Alin+Ahmad) – Led literature review on navigation algorithms.

Programmer (Alin+Ahmad) –

I Wrote and tested line-following

Ahmad has done the coding for the maze.

Engineer (Hossam) – Assembled and adjusted robot for real-world performance.

Expected Challenges and Solutions

Challenges

Solutions

<i>Sensors not detecting the line properly</i>	<i>Adjust sensor placement and sensitivity</i>
<i>Robot moving too fast or unstable turns</i>	<i>Reduce speed, use PID control (if applicable)</i>
<i>Incorrect maze navigation</i>	<i>Improve decision-making logic</i>
<i>Debugging and testing delays</i>	<i>Allocate extra time for troubleshooting</i>

Tools & Resources Required

- LEGO SPIKE Prime Kit

- SPIKE App (Word Block or Python mode)
- Maze layout and test tracks (printed or built with tape)
- Documentation tools (Word, Google Docs, etc.)
- Research articles and tutorials (for algorithm reference)

4.Design process

The design process for this project involved both hardware assembly and software development to ensure the LEGO SPIKE robot could effectively follow a line and navigate a maze. The robot uses **motors, a colour sensor, and a** SPIKE Prime Hub, with programming implemented in Word Block/Python. My primary role was to develop the line-following algorithm, which enables the robot to detect and stay on the black line using sensor feedback and movement adjustments. This section outlines the key design decisions, coding approach, and testing improvements made during development.

My **personal** role was to develop the line following algorithm which I'm going to explain how I did it and afterwards I'm going to share with you the other parts of the designing process.

- The line-following algorithm enables the LEGO SPIKE robot to detect and stay on the black line using a **colour sensor** and motor adjustments. The algorithm continuously reads sensor data and modifies the robot's movement accordingly.

The 1st step was to make the algorithm for the line following, I have used the colour sensor to detect the black line and if the sensor detected black then move forward, if detects white, adjust turning until the black line is found again.

If sensor sees black-Move forward

Else -Adjust turning

2nd step was to detect the obstacle, in this case I used the ultrasonic Sensor and if the obstacle is detected I programmed the robot to stop following the line and execute an avoidance manoeuvre.

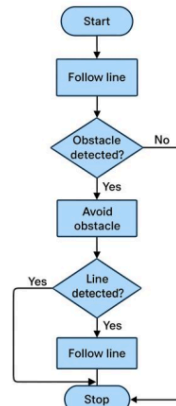
If an obstacle is detected- Stop& turn away

The step 3 is followed by the obstacle avoidance turning left or right, moving a fixed distance to go around and turning back towards the expected line location.

Last step was to regroup on the line. Once the robot is past the obstacle, it must search for the black line again. I used the colour sensor to scan left and right and once black line is found, resume normal line-following mode.

- In the **flowchart** it's a visual explanation which might be easier to understand the overall process of the design.

In the following screengrab it's the code I wrote for the line following and obstacle avoidance. I used Python as I find it much easier to work with and more flexible.





Connect

```
1 from spike import PrimeHub, MotorPair, ColorSensor, DistanceSensor
2 import time
3
4 # Initialize SPIKE Prime components
5 hub = PrimeHub()
6 motors = MotorPair('A', 'B')# Motors connected to ports A and B
7 color_sensor = ColorSensor('C')# Color sensor in port C
8 ultrasonic_sensor = DistanceSensor('D')# Ultrasonic sensor in port D
9
10 # Line-following parameters
11 THRESHOLD = 50# Light intensity threshold (adjust based on environment)
12 BASE_SPEED = 30# Base motor speed
13 TURN_SPEED = 20# Speed when adjusting
14
15 def follow_line():
16     """
17     Moves the robot along the line using a simple if-else method.
18     """
19     while True:
20         light_value = color_sensor.get_reflected_light()
21
22         if light_value < THRESHOLD:# Black detected
23             motors.start_tank(BASE_SPEED, BASE_SPEED)# Move forward
24         else:# White detected (off the line)
25             motors.start_tank(TURN_SPEED, -TURN_SPEED)# Turn slightly
26
27         # Check for obstacles
28         if ultrasonic_sensor.get_distance_cm() < 10:# Obstacle detected within 10 cm
29             avoid_obstacle()
30
31         time.sleep(0.1)
```



Connect

```
32
33 def avoid_obstacle():
34     """
35     Stops, moves around the obstacle, then finds the line again.
36     """
37     motors.stop()# Stop the robot
38     time.sleep(0.5)
39
40     # Move around obstacle (adjust distances as needed)
41     motors.move(5, 'cm', speed=BASE_SPEED)# Move forward slightly
42     motors.move(90, 'degrees', steering=100)# Turn right
43     motors.move(10, 'cm', speed=BASE_SPEED)# Move forward past obstacle
44     motors.move(90, 'degrees', steering=-100)# Turn left
45     motors.move(5, 'cm', speed=BASE_SPEED)# Move forward
46
47     # Resume line-following after avoiding the obstacle
48     find_line()
49
50 def find_line():
51     """
52     Searches for the black line after avoiding the obstacle.
53     """
54     motors.start_tank(-TURN_SPEED, TURN_SPEED)# Rotate left slowly
55     time.sleep(1)
56
57     while True:
58         if color_sensor.get_reflected_light() < THRESHOLD:# Found black line
59             motors.start_tank(BASE_SPEED, BASE_SPEED)# Resume normal movement
60             break
61     time.sleep(0.5)
62 # Start the line-following function
63 follow_line()
```

Besides the line following process and obstacle avoidance there are also other aspects of the process which my teammates contributed to finish the project such as:

Maze Navigation Strategy (Done my teammate Ahmad)

- Detecting intersections and making decisions to navigate the maze.
- Common approaches:
 - Wall-following (Left/Right-hand rule)
 - Predefined path navigation
 - Dead-end detection & backtracking

Key Challenges:

- Ensuring the robot does not get stuck in loops.
- Finding the shortest path to exit the maze.

Software Implementation & Logic

- Developing the algorithm for maze solving.
- Programming the robot using LEGO SPIKE Prime Word Blocks or Python.
- Fine-tuning parameters like motor speed, sensor thresholds, and turning angles.

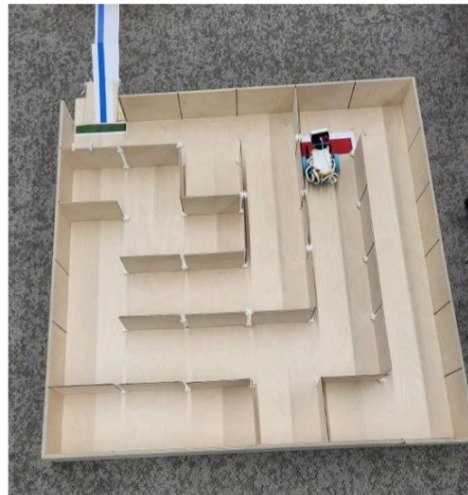
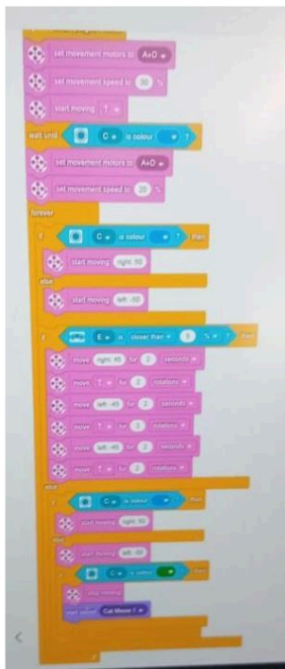
Optimization Techniques:

- Reducing unnecessary movements for faster navigation.

Testing & Iteration

- Running multiple test trials to check performance.
- Adjusting sensor calibration, speed, and turning angles based on observations.
- Identifying and fixing unexpected issues, such as:
 - Overcorrection in line-following.

Maze Navigation Code (Lego spike block)



Robot

Maze Example and Lego spike

6

If I had more time, I would have improved the line-following algorithm by using:

- Two colour sensors instead of one to provide better stability and reduce errors when detecting the line.
- Smoother turning logic by adjusting motor speeds gradually instead of making sharp corrections.
- A more advanced control system like proportional control, which would allow the robot to turn more smoothly rather than zigzagging.
- A line-recovery mechanism, so if the robot loses the line, it will scan left and right to find it again instead of stopping.

5.Evaluation and Results

The purpose of this project was to design a LEGO SPIKE Prime robot capable of following a line, navigating a maze, and avoiding obstacles. To assess the success of each stage of the project, we conducted **multiple test trials**, observing the robot's performance and making necessary adjustments. Below is a breakdown of how each feature was evaluated, the results obtained, and an overall reflection on the project's success.

Assessing Success in Each Stage

To ensure the robot functioned correctly, we assessed each stage separately before combining them into a fully autonomous system.

Line Following Performance

Test Method:



- The robot was placed on a black line track with varying curves and turns.
- The colour sensor was used to detect the contrast between the black line and the white background.
- We observed how accurately the robot adjusted its direction when drifting off the line.

Results:

- The robot successfully followed the line in most cases, adjusting left or right based on sensor readings.

- On gentle curves, the robot responded well, smoothly correcting its path.
- On sharp turns, the robot sometimes struggled, overshooting or momentarily losing the line.
- At high speeds, the robot occasionally zigzagged too much, reducing efficiency.

Evaluation:

- The line-following algorithm worked well, but small refinements were needed for sharper turns.
- Implementing speed adjustments (reducing speed on tight curves) improved performance.
- A dual-sensor setup might have enhanced stability.

Obstacle Avoidance System

Test Method:

- The robot was placed on a track with obstacles in its path.
- The ultrasonic sensor was used to detect objects, and the robot was programmed to stop, turn, and find a clear path.
- We tested how well it rejoined the original line after bypassing an obstacle.

Results:

- The robot successfully detected obstacles and stopped at an appropriate distance.
- The turning mechanism worked, but occasionally, the robot turned too far, making it harder to return to the line.

- In some cases, after avoiding an obstacle, the robot struggled to realign itself with the track.

Evaluation:

- The obstacle detection functioned reliably, but the recovery mechanism after avoidance needed fine-tuning.
 - A potential solution would be to add a searching routine, where the robot briefly scans for the line after avoiding an obstacle.
 - Adjusting the turning angle dynamically based on distance could improve precision.
-

Maze Navigation

Test Method:

- The robot was placed in a simple maze layout with multiple paths.
- It was programmed to follow the line while making navigation decisions.
- Different maze-solving strategies were tested, such as the right-hand rule and dead-end detection.

Results:

- The robot correctly followed the maze path in simpler layouts but struggled with more complex intersections.
- If the maze had dead ends, the robot sometimes got stuck before realizing it needed to backtrack.
- Performance improved after tweaking the decision-making algorithm.

Evaluation:

- The right-hand rule was effective in some cases but had limitations in mazes with loops.
- A more advanced navigation strategy, such as storing previous moves to recognize loops, could improve performance.

- Further optimization could allow the robot to find the shortest path instead of just any path.

The project was largely successful, with the robot being able to complete its tasks under normal conditions. However, several areas could be improved for better performance:

What Worked Well:

- The line-following algorithm was effective in most cases.
- The obstacle detection system functioned reliably, allowing the robot to stop before collisions.
- The robot was able to navigate a maze, making correct turns based on sensor inputs.

Challenges & Areas for Improvement:

- Sharp turns caused occasional delays, requiring better turning adjustments.
- Obstacle avoidance affected line realignment, leading to small navigation errors.
- Maze-solving efficiency could be improved to handle complex intersections better.
- This project was an excellent learning experience, providing insight into robotics, automation, and problem-solving. Despite some challenges, the LEGO SPIKE Prime robot demonstrated the ability to autonomously navigate using sensors and programmed logic. The project highlighted the importance of fine-tuning algorithms, testing extensively, and continuously improving performance. With further refinements, the robot could become even more precise, efficient, and adaptable in real-world navigation scenarios.

This project was an excellent learning experience, providing insight into robotics, automation, and problem-solving. Despite some challenges, the LEGO SPIKE Prime robot demonstrated the ability to autonomously navigate using sensors and programmed logic. The project highlighted the importance of fine-tuning algorithms, testing extensively, and continuously improving performance.

With further refinements, the robot could become even more precise, efficient, and adaptable in real-world navigation scenarios.

"Apologies for the image quality " Thank you!

Alin Astefanei (2404824)

Integration Project

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

27 /100

PAGE 1

PAGE 2



Comment 1 | Transferable Skills(15)

basic understanding of the task, but could be improved by clearly stating the challenge with LEGO Spike Prime and its broader purpose in STEM learning.



Comment 2 | Engagement with Literature(15)

in text citation should follow IEEE format

PAGE 3



Comment 3 | Engagement with Literature(15)

Some understanding of programming logic is evident, but no literature or tools review was included. exploring credited educational sources relevant to LEGO Spike and robotics to strengthen the research section could have been done.

PAGE 4



Comment 4 | Transferable Skills(15)

clearer roadmap with a time plan and breakdown of team responsibilities (especially your own) is expected

PAGE 5

PAGE 6



Comment 5 | Knowledge and Understanding(40)

Your design logic is well-structured in Python, however, the brief required the use of Word block programming with screenshots from the LEGO Spike environment. Hardware design is not fully detailed either. Please revisit the design process using visual blocks to better align with project goals.

PAGE 7

PAGE 8

PAGE 9

**Comment 6** | **Transferable Skills(15)**

the figure could have been numbered and labelled

**Comment 7** | **Cognitive and Intellectual(30)**

Evaluation is currently missing. reflection on what worked, what didn't, and how you tested your design is missing. Screenshots, test results, and a brief comparison between expected and actual outcomes would make this section much stronger.

GRADING FORM: DC-ACHA4008

ALIN ASTEFANEI**27****ENGAGEMENT WITH LITERATURE(15)****3****KNOWLEDGE AND UNDERSTANDING(40)****16****COGNITIVE AND INTELLECTUAL(30)****4****TRANSFERABLE SKILLS(15)**

