Final Report

Prepared by:
Sachin Daya
Keturah Naidoo
Group 42

Prepared for:
EEE3099S and EEE3097S
Department of Electrical Engineering
University of Cape Town

October 22, 2024

Declaration

1. We know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. We have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed and has been cited and referenced.
3. This report is our own work.
4. We have not allowed, and will not allow, anyone to copy our work with the intention of passing it off as their own work or part thereof.

Sachin Daya

October 22, 2024

Keturah Naidoo

October 22, 2024

Contents

Executive Summary

1.1 Project overview

This project entailed the design and development of an autonomous micromouse robot. Using sophisticated sensor-based localization, movement control, and system optimization techniques, the micromouse's main objective was to navigate through a maze. Additionally, the project was developed around four major milestones, each of which focused on a crucial stage of development, ranging from sensor calibration and early system design to autonomous navigation and performance improvement. Several technical obstacles were addressed through this project, namely, sensor accuracy, power management, and real-time processing. Subsequently, creative engineering solutions were later developed to address these problems, enabling the micromouse to navigate precisely and select the best path.

1.2 Milestone requirements

This project was developed around the following key milestones.

Table 1.1: Micromouse Milestone Requirements

Milestone	Description	Deliverables	Deadline
Milestone 1	Initial design	Status report outlining subsystems and their integration.	5th August 2024
Milestone 2	Sensor Calibration and Localization	Demonstration of wall sensing and line following capabilities.	11th September 2024
Milestone 3	Autonomous Navigation	Demonstration of navigation and mapping.	6th October 2024
Milestone 4	System Optimization	Optimized system performance.	18th October 2024

1.3 Sensor subsystem functionality and operation

The micromouse's sensor subsystem utilizes infrared (IR) LEDs for detecting both maze walls and line following. Infrared light is continuously emitted by the IR LEDs, and the system detects the reflected signals to calculate alignment with the line and proximity to walls. As the micromouse gets closer to an impediment, the signal is strengthened by side-facing and front-facing infrared LEDs, which helps it detect adjacent walls and change course to prevent collisions. To ensure that the micromouse remains centered for line detection, the system measures the strength of reflection and modifies the orientation to compensate for drift. The microcontroller processes this sensor data to make judgments about movement and navigation in real time. It can modify its course to stay on course and steer clear of obstacles by identifying threshold values. This constant feedback loop between the sensors and the navigation system is essential for accurate movement and overall maze-solving effectiveness since this keeps the micromouse aligned and steers clear of walls whilst mapping the maze. Furthermore, the efficiency of the sensor subsystem is adjusted throughout the optimization process to increase detection accuracy, which facilitates quicker navigation and lowers error rates, ultimately improving the micromouse's capacity to successfully navigate the maze.

1.4 System interfacing

To illustrate the relationship between the micromouse's key subsystems, the system architecture diagram below highlights the interactions between the sensor subsystem, microcontroller, motor drivers, and power system. This high-level view demonstrates how sensor data is processed by the microcontroller to control motor movements and navigate the maze effectively.

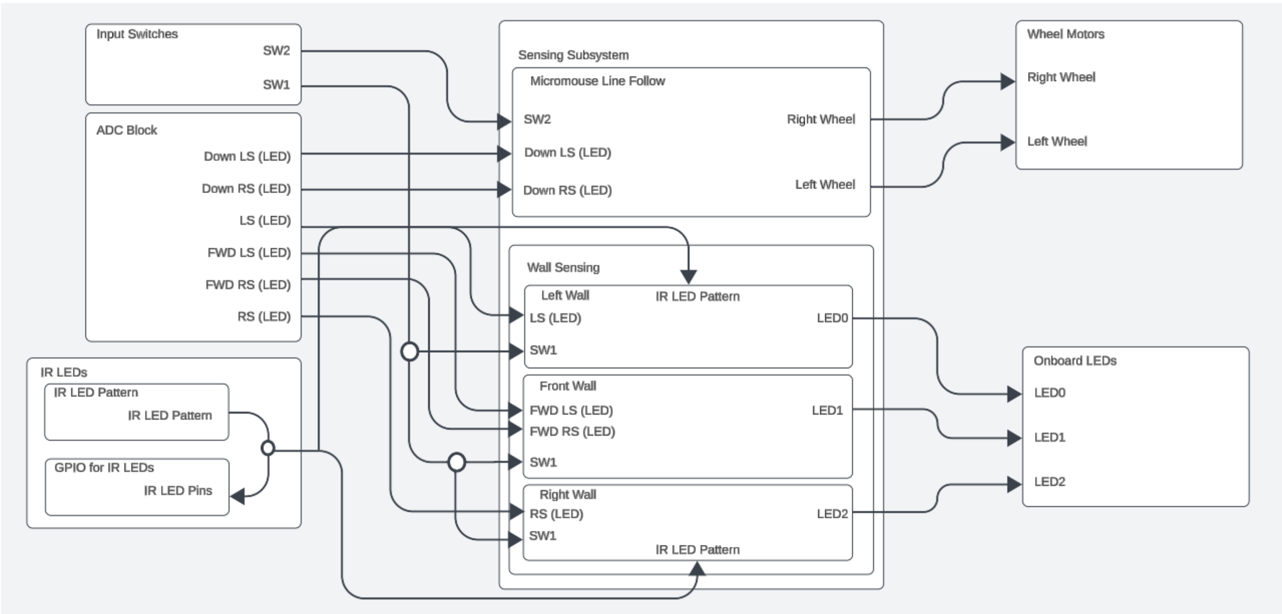


Figure 1.1: Interfacing diagram of sensing subsystem with the rest of the micromouse.

Requirements Analysis

2.1 Summarised requirements

The project's scope encompasses the design, implementation, and testing of a sensor subsystem using infrared LEDs to enable line following and wall detection, navigation, mapping and optimization. The following table details all the project requirements.

Table 2.1: Micromouse Requirements

ID	Requirement Description
RQ1	Micromouse must sense a wall.
RQ2	Micromouse must follow a line.
RQ3	Micromouse must be able to make a turn depending on the input from the sensors.
RQ4	Micromouse must stop when a wall is in front of it.
RQ5	Micromouse must mitigate the effect of ambient light.
RQ6	Micromouse must navigate through the maze and map it.
RQ7	Micromouse performance must be optimized.

2.2 Limitations and external factors

The limitations due to external and internal factors need to be identified to create an effective solution for navigating through the maze and ensuring optimized performance .

Table 2.2: Internal and External Factors Affecting Micromouse Navigation

ID	Factor Type	Brief Description
IF1	Sensor Accuracy	Proper calibration and precision of sensors for detecting walls and obstacles.
IF2	Emitter Accuracy	The accuracy of when the emitters are turned on and off with the PWM.
IF3	Algorithm Efficiency	The efficiency of the path-planning algorithm in processing sensor data.
IF4	Inertia and Speed	The micromouse's physical momentum can lead to overshooting turns or walls.
IF5	Processing power	Limited microcontroller speed could slow down data processing and decision-making.
IF6	Battery power consumption	Power management must ensure sufficient battery life for navigation and sensor use.
EF1	Maze Layout	Complexity and geometry of the maze affecting path-planning decisions.
EF2	Lighting Conditions	Ambient lighting potentially interfering with the IR sensors.

Design and Implementation of Navigation

3.1 Summarised requirements

For milestone three, the focus is on ensuring that the micromouse can actively navigate the maze, making decisions and executing turns based on the calibration and initialization routines established in milestone two. This stage builds upon the foundation where the micromouse was calibrated, identified and held a line, and developed surrounding awareness. In milestone three, the micromouse is expected to move smoothly through the maze, detecting walls or paths, and making turns accordingly. The micromouse must process sensor inputs in real-time to make decisions on whether to continue straight, turn left or right, or stop or in some cases do a 180° rotation. This requires integrating the sensor feedback from milestone two with decision-making algorithms that guide the mouse's movement.

3.2 Limitations from external and internal factors

Below are the limitations due to the external and internal factors mentioned in Table 2.2.

IF1: Sensor Accuracy

The ability of the micromouse to detect walls and obstacles is highly dependent on the accuracy of its IR sensors. Proper calibration is critical to ensure that the sensors provide reliable and consistent distance measurements. Without accurate sensor readings, the micromouse may fail to detect walls or misjudge distances, leading to collisions or inefficient navigation paths. Environmental factors such as temperature, dust, or electrical noise may also introduce inaccuracies, making it important to regularly re-calibrate the sensors and account for these variables.

IF2: Emitter Accuracy

Emitters need precise timing to activate and deactivate in sync with the pulse width modulation set in the micromouse template provided. Any delay or inconsistency in turning the emitters on or off can lead to skewed sensor readings, causing the micromouse to misinterpret the maze layout. Proper synchronization between the emitters and sensors ensures that the micromouse receives valid data about its surroundings. Poor emitter accuracy can distort obstacle detection, especially in scenarios where precise measurements are required, such as detecting nearby walls.

EF1: Maze Layout

The complexity and structure of the maze play a significant role in how the micromouse approaches navigation. Mazes with multiple dead-ends, narrow passages, and sharp turns present a greater challenge, requiring advanced path planning and maneuverability. Simple maze layouts with fewer branches or wide turns allow for more straightforward navigation, whereas intricate designs demand more sophisticated algorithms and better control of the micromouse's movement. The layout may also include deceptive elements like loops or misleading paths that can slow down navigation.

EF2: Lighting Conditions

Lighting conditions, particularly ambient light, can interfere with the performance of IR sensors used by the micromouse. IR sensors emit light at specific wavelengths to detect obstacles, and external light sources, particularly sunlight or bright artificial lights, can cause interference or saturation in the sensor readings. This can result in false positives or missed detection of walls and obstacles. Shielding the sensors or calibrating them to distinguish between the micromouse's own IR signals and ambient light can help mitigate this issue, but ambient conditions can still introduce variability in sensor performance.

3.3 Design thoughts and initial solutions

3.3.1 Line following

From Milestone 2, the micromouse needed to follow a line, the micromouse could not detect an intersection at that point. It needs to detect an intersection so that it knows when to stop and rotate. The initial design was to use the down_LS and down_RS sensors to identify a line when both were less than the threshold set when calibrating (from milestone 2 calibration was done when both sensors were not placed on the line).

That was not working effectively, this may be caused by the sensor accuracy mentioned in IF2, so the sensors that were placed facing the wheels (mot_LS and mot_RS) were turned down and used for the detection of an intersection since they were on the far right and left sides of the sensing board they were constantly kept on as it did not affect the other sensors functionality.

Due to the effect of ambient light, the line following functionality becomes inconsistent, an ambient light algorithm was introduced to try to mitigate this effect outlined in EF2. This algorithm was used to check the ambient light by turning off all the emitters and then measuring the light received by the sensors, this value is then subtracted by the threshold value originally calibrated when the sensor is on, this can be seen as the ambient difference, this value is then scaled by 0.9 and used as the new measure against what the sensor will receive, this idea is then implemented at each intersection to re-calibrate the ambient light. The main algorithm is shown below in Figure 3.1

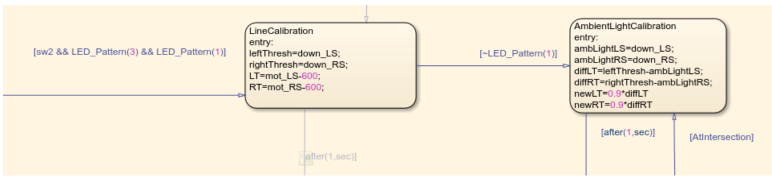


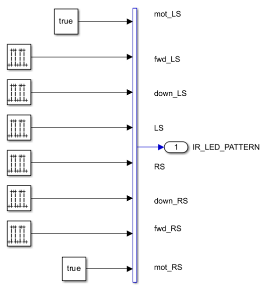
Figure 3.1: Ambient Light mitigation algorithm.

3.3.2 Wall sensing

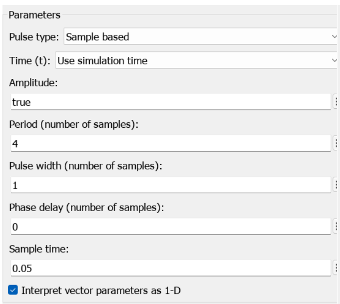
From Milestone 2, we failed to meet the requirement of surrounding awareness. The micromouse could not stop at a wall and the wall sensing did work but was flawed sometimes based on where in the maze the micromouse was placed, this was caused by the lighting conditions highlighted in EF2. The internal factor that could also be affecting the surrounding awareness is IF2. The emitter accuracy was very important to understand to provide a good solution for sensing the environment of the maze. Since the emitters were turned on periodically using a pulse width modulator. In milestone 2 this was done using a time-based PWM. If you analyze the sensing board you can see that the forward-facing sensors (fwd_LS and fwd_RS) and the side-facing sensors (LS and RS) interfere with each other respectively. This is why the emitter accuracy is important if the emitters are interfering with each other the light received by the sensor will not be reflective of what is happening in the maze.

Thus to resolve this issue instead of using a time-based PWM, we first tried to turn the emitters on and off manually to get more accurate pulses, as an issue with the time-based PWM was the phase delay in which each emitter was set was not reflective of what was happening on the sensor board, this was confirmed by running the debug mode.

Initially, it was thought that turning the emitters on and off manually would solve this but still, the pulses were not occurring at the correct time, issues also developed in the code and logic being too complicated to integrate with the rest of the micromouse's functions. Thus we thought about using a sample-based PWM, this turned out to be much more accurate. Figure 3.2 shows the configuration for the emitters.



(a) Configuration of how the emitters will be turned on.



(b) Configuration of the sample based PWM.

Figure 3.2: IR Led Pattern configuration

To resolve the issue experienced by the lighting conditions, the same algorithm for detecting the ambient light is mentioned in the line following the design section and shown in Figure 3.1. The idea is just translated to detecting a wall instead of a line.

To help with both line following and wall sensing, the sensors were covered with insulation tape to reduce the amount of leakage light they received. This made the sensors more accurate and stopped the system from giving false positive readings.

3.3.3 Making turns

To allow the mouse to make a turn it needs a combination of both line following and wall sensing. It should only make a turn when on an intersection and it should only turn in the direction of where there is no wall. A timing method was used for the rotations, rather than the gyroscope. The gyroscope would have been an effective method to use as it would reduce the overshoot or undershoot caused by using the timing method, as the rotation time would change based on how much battery level the micromouse has at that moment, the less battery the more time is needed to get to the same point. Also, another point to mention is that since the mouse does not sit directly on the intersection when it is detected it needs to be adjusted forward, this was also implemented using a timing method, in which the same issue with the rotations occurs. The effects of battery consumption and inertia and speed of the micromouse are discussed in chapter 4. Figure 3.3 shows the transitions are done after a certain amount of time.

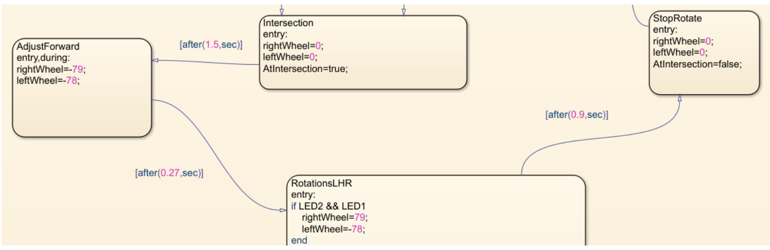


Figure 3.3: Showing the implementation of rotating to make a turn in the maze.

The gyroscope could not be implemented as there were troubles with the calibration of the mouse. If the method was used here is a brief explanation of how it would have been implemented.

Using a gyroscope to make a micromouse rotate on the spot and execute turns in a maze is an effective way to ensure precise navigation. The gyroscope measures the angular velocity of the micromouse as it rotates, helping it maintain awareness of its orientation within the maze. To make a turn, the micromouse must be programmed to rotate by a specific angle, such as 90° for a right or left turn, or 180° for reversing direction.

The first step is to calibrate the gyroscope. This is necessary to eliminate any drift or bias that can affect the accuracy of the readings. Calibration involves establishing a zero-rate offset when the micromouse is stationary. After calibration, the micromouse can accurately measure the angular velocity as it rotates on the spot. The rotation is achieved by driving the left and right motors in opposite directions.

As the micromouse rotates, the gyroscope measures the rate of rotation around the z-axis. The angular velocity data from the gyroscope is integrated over time to calculate the total angle of rotation. Once the integrated angle matches the target rotation angle the motors are stopped, and the micromouse completes its turn. This method ensures that the micromouse can rotate accurately and make precise turns in the maze.

3.3.4 Navigation through the maze

For basic navigation through the maze, the wall-following method was used, more specifically the left-hand rule. It works simply by keeping one hand in contact with one wall of the maze (in this case the left sensor). The micromouse is assured not to become lost and will find an alternative exit if one exists; otherwise, the algorithm will return to the entrance after exploring every corridor adjacent to that connected wall section at least once. This approach follows a depth-first, in-order tree traversal method.

This method does have some disadvantages if the maze has islands (where walls are not connected to the outer boundary), the micromouse may get stuck going around in circles on a loop, never reaching the exit. It is not an optimal solution, as it does not find the shortest path a solution for this is discussed in chapter 4. If the maze is not simply connected some issues might occur in navigation as the micromouse might end up in a loop and not find the exit at all. Thus the main factor affecting this function is EF1 the maze layout, this issue gets mitigated when doing the optimization algorithm for the micromouse. Figure 3.4 shows how the left-hand rule algorithm is implemented. The LEDs are used to indicate whether a wall is present or not, if it is true there is a wall on that respective side.

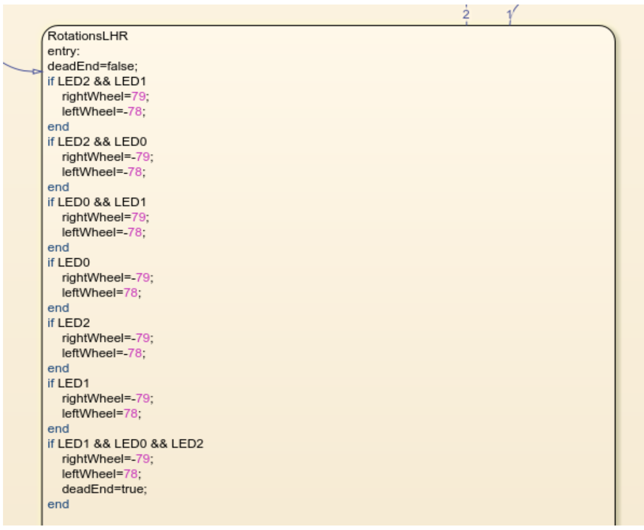


Figure 3.4: Left-hand rule implemented in the rotation of the micromouse.

3.4 Ranking contingencies

Multiple factors affected the overall navigation of the maze, mainly the ambient light was the biggest challenge, the next issue was the sensors not being accurate enough, and the emitters also providing inaccuracies and the layout of the maze might have caused an issue for navigation if it is very complex.

Table 3.1: Contingency ranking for micromouse design.

Contingency	**Description**	**Priority**
EF2: Lighting Conditions	Ambient light interferes with IR sensors, leading to inaccurate readings. An ambient light algorithm adjusts sensor thresholds dynamically to mitigate this issue.	Highest
IF1: Sensor Accuracy	Accurate sensor calibration is essential for detecting walls and obstacles. Regular calibration is required.	High
IF2: Emitter Accuracy	Precise synchronization of emitters and sensors is critical for accurate obstacle detection. Misaligned pulses can lead to distorted readings.	Medium
EF1: Maze Layout	The complexity of the maze affects navigation. Advanced path-planning algorithms can handle intricate layouts with dead-ends or loops.	Low

3.5 Final Design

A selection criterion was created to determine the best solution for navigation of the maze. Table 3.2 shows this criteria.

Table 3.2: Selection Criteria and Limitation Reduction

Criterion ID	**Criterion Description**	**Related Requirement(s)**	**Limitation Addressed**
C1	Wall Detection Accuracy: The solution must ensure the micromouse reliably senses walls using its sensors to navigate.	RQ1, RQ4	IF1: Sensor Accuracy; IF2: Emitter Accuracy
C2	Line Following Capability: The micromouse must effectively follow a line using its sensors, especially at intersections.	RQ2	IF1: Sensor Accuracy
C3	Turning Responsiveness: The solution should allow the micromouse to make turns accurately based on sensor input without overshooting.	RQ3	IF1: Sensor Accuracy; EF1: Maze Layout
C4	Obstacle Avoidance: The micromouse must stop immediately when detecting a wall directly in front of it to prevent collisions.	RQ4	IF1: Sensor Accuracy
C5	Ambient Light Mitigation: The solution must address the impact of ambient light on sensor performance to ensure reliable data in various lighting conditions.	RQ5	EF2: Lighting Conditions
C6	Maze Navigation: The micromouse should navigate through the maze effectively while creating a map of its surroundings for optimal path finding.	RQ6	EF1: Maze Layout

The following design choices were made for the final design to fulfill the requirements as best we can. To meet criterion **C1**, the ambient light detection solution was implemented, it also met criterion **C5** and **C4**. The left-hand rule for solving a maze was implemented to meet criterion **C6**. The motor sensors were used to detect an intersection to meet criterion **C3**. For criterion **C2** to be met, the ambient light solution was implemented along with taping the IR sensors to prevent detecting leakage from the other emitters.

3.6 Results and Testing

The following acceptance tests were conducted as shown in Table 3.3

Table 3.3: Acceptance Tests for the Micromouse Navigation System

Test ID	Test Description	Requirements Fulfilled	Pass Criteria	Fail Criteria
AT01	Wall Detection: The micromouse must stop when a wall is detected.	RQ1, RQ4	The micromouse stops before hitting the wall.	The micromouse collides with the wall or does not stop.
AT02	Line Following: The micromouse must follow a line accurately.	RQ2	The micromouse follows the line without deviating.	The micromouse strays from the line or loses track.
AT03	Turning at Intersections: The micromouse must make a turn when it detects an intersection.	RQ3	The micromouse turns accurately based on sensor input.	The micromouse misses the turn or makes incorrect turns.
AT04	Ambient Light Mitigation: The micromouse must adjust for ambient light interference and maintain functionality.	RQ5	The micromouse performs consistently in varying lighting.	The micromouse's sensors give inaccurate readings in different lighting conditions.
AT05	Maze Navigation: The micromouse must navigate the maze and avoid dead ends or loops.	RQ6	The micromouse navigates through the maze efficiently, mapping it without getting stuck in loops.	The micromouse fails to complete the maze or becomes stuck in a loop.

3.6.1 Testing procedure

AT01

Build the code, place the micromouse in the maze, and create a path with a dead-end, let the mouse follow the path and see how it reacts at the dead end.

AT02

Build the code, place the micromouse on the line and let it calibrate, connect the mouse to the debugger, analyse sensor inputs, and see if it follows the line after calibration.

AT03

Build the code, and place the micromouse in the maze. Test if the micromouse detects an intersection and records the walls at that point, then record if the micromouse turns in the right direction depending on conditions at that point.

AT04

Check if AT01 to AT03 is passed. Connect the micromouse mouse to the debugger and see the effects on the sensor when the mouse is placed in different parts of the maze or different environments. The testing should be done at different parts of the day.

AT05

Place the micromouse in the maze and see if it calibrates correctly, makes turns, and follows a line. Check if AT01 to AT04 is passed.

3.6.2 Results

After following the testing procedure the following results were gathered. The data will be from readings of the IR sensor values when placed in different conditions. Table 3.4 shows the data from the debug mode. Note all the data is in volts (V).

Table 3.4: Sensor Data Table

Sensor	Left Wall	No Wall	Right Wall	Front Wall	Line	No Line
mot_LS	3.3	3.3	3.3	3.3	2.5	3.3
fwd_LS	3.3	2.43	3.3	1.23	3.3	3.3
LS	1.23	0.83	0.31	2.28	0.3	0.24
down_LS	2.73	1.10	1.04	3.01	0.89	1.05
down_RS	2.01	1.05	0.97	3.07	0.85	1.23
RS	0.48	0.73	0.87	1.05	0.54	0.48
fwd_RS	3.3	2.47	3.3	0.83	3.3	3.3
mot_RS	3.3	3.3	3.3	3.3	2.52	3.3

The results shown in Table 3.4 take the readings from one round of testing, this was done multiple times to reduce any uncertainties and allow for the most accurate results.

The data was analyzed and the micromouse was tested in the maze, to summarise the line following and intersection detection were working as expected and the effect of ambient light was mitigated as well as the effect of the emitter and sensor accuracy displaying that the design decisions made were to good effect.

The wall detection was not as successful the results were inconsistent and the ambient light played a large role in the functionality of our algorithm. It worked for the most part (75% of all tests), but the underlying functionality was there meaning it sensed a wall and stopped when a front wall was detected.

The mitigation of ambient light was overall successful and the micromouse was more consistent in following a line and detecting a wall than in milestone 2. Sometimes it did not work so there is still room for improvement.

The ability for the micromouse to turn was not so successful since a timing method was used, there was a lot of overshoot, and the time to complete a rotation changed depending on the charge of the battery therefore, turns were not as accurate and the line was missed to continue solving the maze. The ambient light made the micromouse turn into a wall as the sensor received a false positive.

Maze navigation was not successful as the micromouse could not make turns. This was a combination of the effect of ambient light and a poor design decision in using a timing method and not the gyroscope. Also the Left-hand rule for maze solving was not as effective as the maze layout was sometimes complex with loops and dead ends.

From the results gathered using the debugging and testing the micromouse this is how the acceptance tests were passed and failed.

Table 3.5: Results after performing acceptance tests.

Test Procedure	**Pass/Fail**
AT01	Pass
AT02	Pass
AT03	Fail
AT04	Pass
AT05	Fail

In conclusion, the micromouse was temperamental and not consistent enough, overall milestone 3 did see some improvements made on milestone 2 but was not passed as the micromouse failed tests AT05 and AT03 which directly relates to the navigation of the maze.

Refer to Appendix A for screenshots from the debug mode and images of the final micromouse and the maze used for testing.

Design and Implementation of Optimization

4.1 Summarised requirements

Following the development and testing of the micromouse's initial navigation capabilities, Milestone 4 concentrated on enhanced performance. The objective at this point was to optimize the micromouse's navigation around the maze in order to minimize mistakes and shorten the time taken to complete the maze. This entailed enhancing the path-planning algorithm for faster decision-making, honing the sensor subsystem for more precise wall and line recognition, and optimizing the micromouse's movement control to prevent collisions and overshooting turns. To ensure the micromouse can function within the limitations of its battery life, power management is also adjusted. While the micromouse failed to be optimized by the time of the demo for this project, this section covers how optimization would of been implemented.

4.2 Limitations from external and internal factors

Below are the limitations to optimization due to the external and internal factors mentioned in Table 2.2.

IF3: Algorithm Efficiency

The speed and effectiveness of the micromouse's labyrinth navigation are directly impacted by the effectiveness of its path-planning algorithm.Therefore, to enable real-time decision-making, algorithms such as flood-fill and strictly left or right-hand rule must be executed with the least amount of processing overhead possible. Additionally, the micromouse's responses to sensor inputs may be slowed down by inefficient algorithms that lead to prolonged processing times. Furthermore, the micromouse may travel longer routes, run into dead ends more frequently, or retrace its steps needlessly if the algorithm is unable to optimize the route, all of which could reduce the overall performance.

IF4: Inertia and Speed

Inertia makes it difficult for the micromouse to stop suddenly or perform rapid turns. The micromouse's momentum may cause it to overshoot turns or fail to stop in time as it approaches a wall when traveling at quicker speeds. The control and speed of the micromouse must be balanced in order to counteract this. This can be done by adding acceleration and deceleration processes to its movement logic. Increased speed can speed up the time taken to solve the maze, however, if inertia is not accounted for, the micromouse might crash or become stuck.

IF5: Processing Power

Optimization can be greatly impacted by a microcontroller's limited processing power since it slows down data processing and decision-making. Navigation and obstacle avoidance delays may result from the microcontroller's inability to process sensor input and make real-time modifications rapidly enough. Additionally, the micromouse may take longer to determine the best pathways or react to changes in the environment as a result of ineffective path-planning, which could lengthen the time needed to complete the maze. Further impairing system efficiency is the microcontroller's incapacity to manage complicated algorithms effectively, which may necessitate the employment of less ideal, simpler solutions. Therefore, despite limited processing power, effective optimization necessitates balancing the computational load to prevent delays in crucial operations like sensor processing and movement control.

IF6: Battery Power Consumption

Efficient power management is crucial to ensure the micromouse can operate for long enough to complete the maze. High power consumption, particularly from sensors, motors, and the microprocessor, can quickly drain the battery. Power-intensive components should be used carefully to maximize performance; sensors and motors should only be turned on when absolutely required, and the microcontroller should handle tasks with as little power as possible. Furthermore, when there is little activity or demand, power-saving modes must to be used. For the micromouse to be able to navigate the maze and save enough energy to maintain operation, energy use and functionality must be balanced.

4.3 Optimization design

The following table outlines optimization methods that were considered.

Table 4.1: Optimization methods

ID	Optimization Method	Brief Description
OM1	Flood Fill Algorithm	Uses a maze-solving strategy to explore all possible paths and determine the optimal route to the centre of the maze.
OM2	Left-Hand Wall Following	The micromouse strictly follows the left-hand wall, ensuring continuous movement and avoiding getting lost.
OM3	Dead-End Detection	Identifies dead-ends early and quickly re-routes the micromouse to prevent wasted exploration time.
OM4	Path Memory	The micromouse remembers previously explored paths to avoid redundant exploration and minimize errors.
OM5	Speed Control Optimization	Adjusts speed dynamically based on the proximity to walls or sharp turns, preventing overshooting and maintaining control.

OM1: Flood Fill Algorithm

The flood fill algorithm is a sophisticated maze-solving technique where the micromouse methodically investigates each path and goes backwards to fill the maze grid with information. It subsequently employs the studied paths to find the fastest and shortest route to the destination. Compared to more straightforward approaches, this strategy uses more computer resources but is effective at identifying the best paths. There was not enough time to fully implement this method therefore it was not utilized,despite being the best method.

OM2: Left-Hand Wall Following

The left-hand wall-following method is a simple strategy where the micromouse continuously keeps its left sensor aligned with the wall. The micromouse constantly follows the left wall and turns left where possible to navigate through the maze. Where a left turn is not possible the micromouse moves foward or right. While this method allows for easy navigation through the maze, it is not as fast as flood filling.

OM3: Dead-End Detection

Early detection of dead ends enables the micromouse to promptly eschew pointless investigation. The micromouse can increase its total maze-solving speed by detecting these dead-ends with the use of its sensor data, which allows it to make quicker judgments and avoid wasting time.

OM4: Path Memory

The micromouse can save information about previously explored paths thanks to path memory optimization. By eliminating unnecessary exploration, this helps it to avoid going back the same way and can drastically cut down on the amount of time needed to complete the maze.

OM5: Speed Control Optimization

The micromouse's speed is adjusted using dynamic speed control, which slows it down close to obstructions or sharp corners and speeds it up in open spaces. This keeps it from overshooting corners or running into walls while maintaining precision and speed.

4.4 Optimization Results

While various optimization techniques discussed in Table 4.1 were considered, only one was implemented. This was due to the navigation component being incomplete ,making optimization difficult. The one optimization technique that was implemented was left-hand wall following which enabled the micromouse to continuously follow the left wall and turn left where possible. This technique enabled the micromouse to traverse the maze, albeit in a less efficient manner.

This approach was far from ideal, even if it guaranteed the micromouse could make its way through the maze. Despite being easy to use, the left-hand wall-following method is fundamentally slower and less dependable than more sophisticated techniques such as the flood fill algorithm. Particularly in intricate mazes with loops or isolated parts, left-hand wall following tends to cover more of the labyrinth needlessly and does not ensure the quickest or shortest path to the destination. Because of this, the micromouse's motions were not streamlined for efficiency or speed, and it took longer to reach the objective.

Additionally, the micromouse's performance was directly impacted by the lack of usage of more sophisticated optimization strategies like flood filling, which could have greatly increased navigation efficiency by examining every path and choosing the shortest one. Flood filling would have reduced the number of unnecessary moves and accelerated labyrinth completion times by enabling the micromouse to dynamically evaluate and select the best routes.

Therefore, the micromouse failed to fulfill the performance requirements for Milestone 4 because of its insufficient optimization. The micromouse's speed, precision, and energy efficiency could not be fully optimized due to its unfinished navigation component and dependence on a less effective method. Consequently,it was difficult to live up to the final phase's expectations, underscoring the necessity of additional improvement and the application of enhanced optimization techniques in subsequent iterations.

Conclusion

This project successfully delivered a functional micromouse, capable of basic navigation through a maze. Key milestones, including sensor calibration, autonomous navigation, and system optimization, were completed with varying degrees of success. Despite encountering challenges such as sensor accuracy, ambient light interference, and inconsistent motor control, significant progress was made in overcoming these technical issues. While the micromouse met some of its functional requirements, such as line following and partial wall detection, it struggled with more complex tasks like precise turning and optimal navigation through the maze.

The overall performance and maze-solving efficiency were affected by the restrictions in the existing design, which included the left-hand wall-following method and the dependence on a timing-based turning mechanism. Even though the micromouse showed little progress at each milestone, it was not entirely able to solve a maze on a regular basis. To overcome these constraints, more iterations are required, particularly in the control and optimization subsystems.

5.1 Recommendations

- Improved Sensor Calibration:In order to improve wall recognition and line following accuracy and reduce mistakes brought on by ambient light or sensor misalignment, future iterations should concentrate on improving the sensor calibration procedure.
- Gyroscope Integration for Turning: The micromouse's ability to make exact turns without overshooting, particularly in narrow maze corners, could be greatly enhanced with the inclusion of a gyroscope.
- Advanced Pathfinding Algorithms: The micromouse's capacity to discover the shortest path and minimize needless investigation might be significantly enhanced by using more effective maze-solving algorithms, such as flood fill.
- Power Management Optimization: To increase battery life and guarantee steady performance, the micromouse's power consumption should be optimized, especially with regard to controlling motor speeds and sensor functions.
- Robust Testing and Debugging: It is advised to conduct ongoing testing in a variety of lighting scenarios and labyrinth layouts in order to uncover and resolve unforeseen problems early in the design process and guarantee dependable obstacle detection and navigation.

  Future developments can improve the micromouse's performance and guarantee more seamless, effective navigation through progressively intricate mazes by concentrating on these areas.

Appendix

A.1 References

[1] Wikipedia Contributors, "Maze-solving algorithm," Wikipedia, Jul. 23, 2024.
https://en.wikipedia.org/wiki/Maze-solving_algorithm

[2] GeeksforGeeks Contributors, 'Flood Fill Algorithm,' GeeksforGeeks, Jul. 21, 2024.
https://www.geeksforgeeks.org/flood-fill-algorithm/

[3] "ReferenceEntities," Mathworks.com, 2024.
https://www.mathworks.com/help/matlab/

A.2 Micromouse results and debugging information

A.2.1 Final micromouse

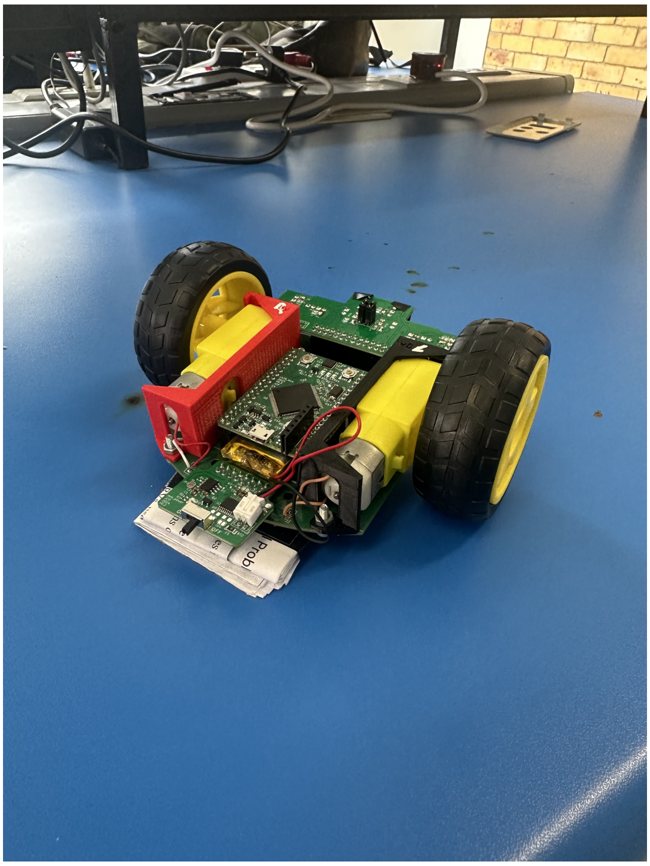


Figure A.1: Micromouse

Figure A.2: Underneath of the micromouse.

A.2.2 Testing the micromouse

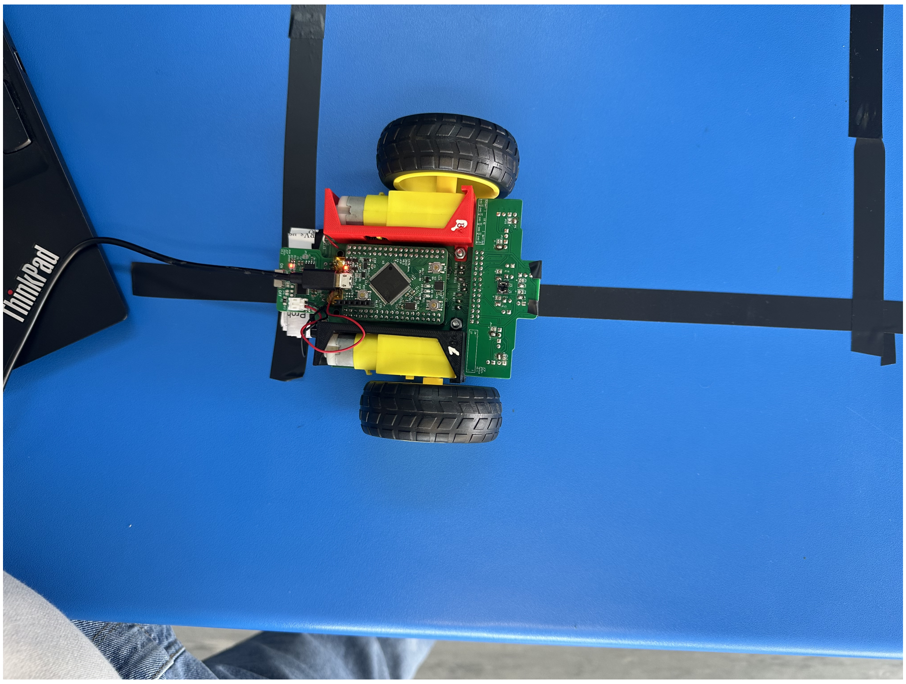


Figure A.3: Line following testing

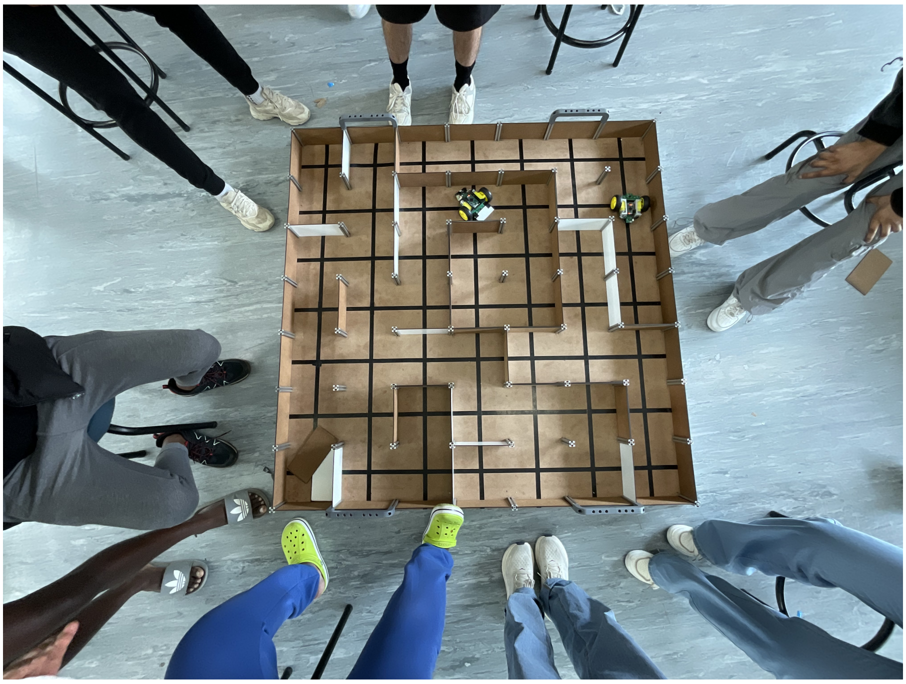


Figure A.4: Maze used to test the micromouse and demo.

A.2.3 Debugging the micromouse

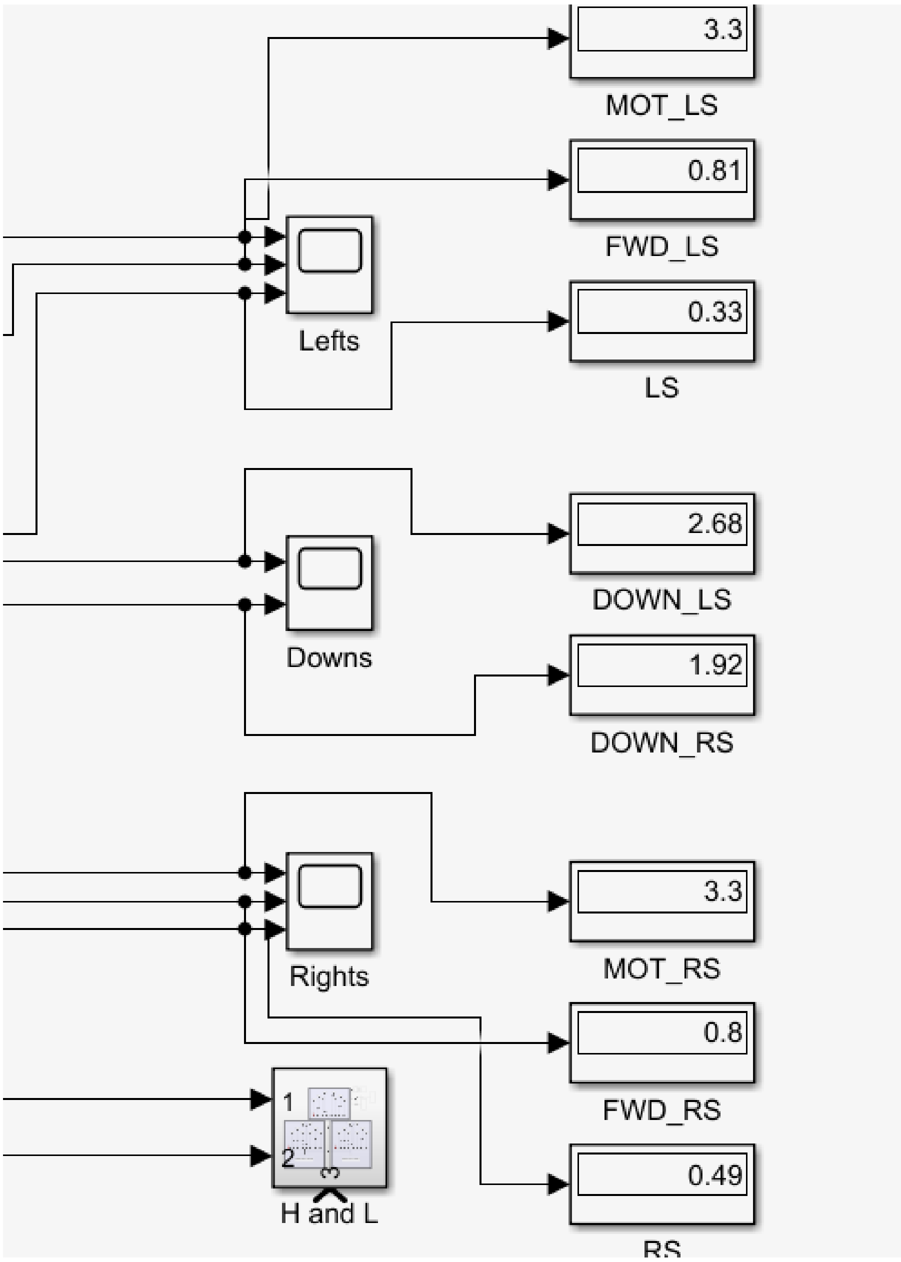


Figure A.5: Testing with front wall present.

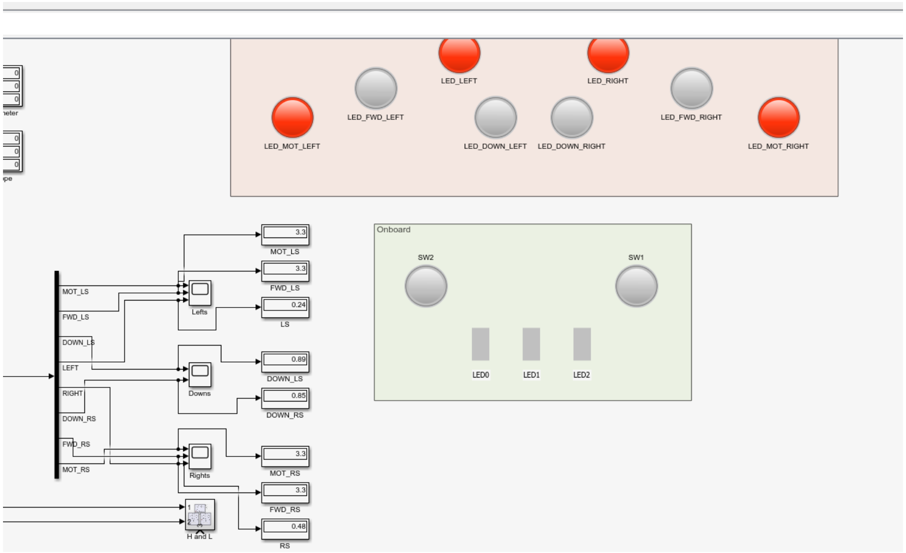


Figure A.6: Testing with left wall and no line present.

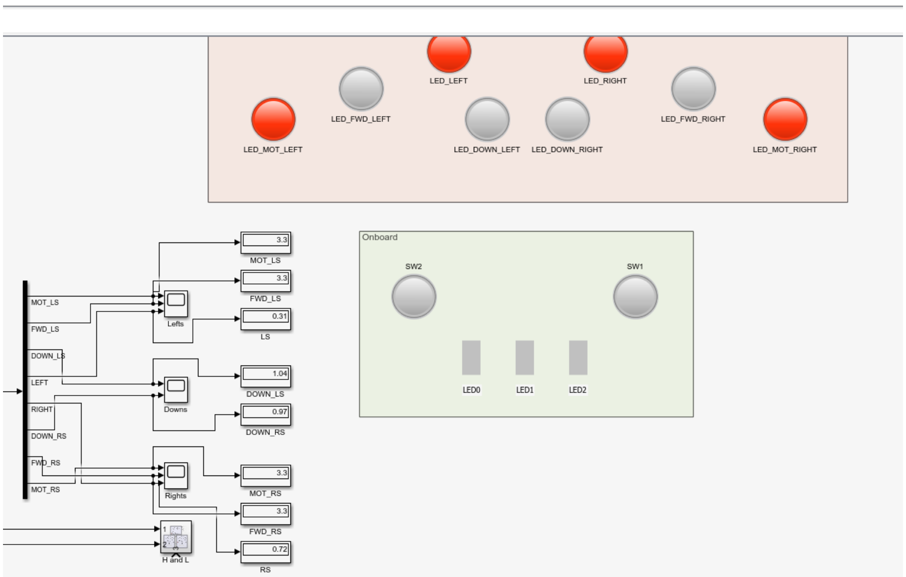


Figure A.7: Testing with right wall and no line present.

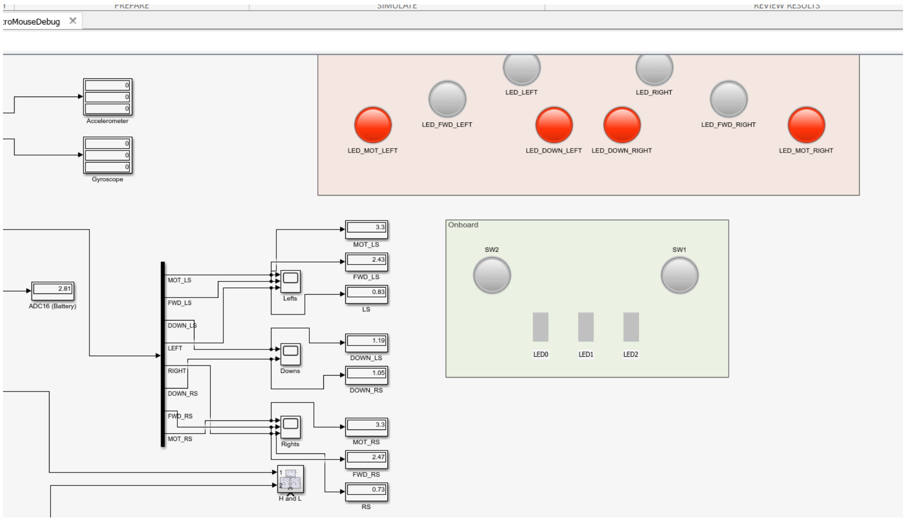


Figure A.8: Testing with no wall and no line present.

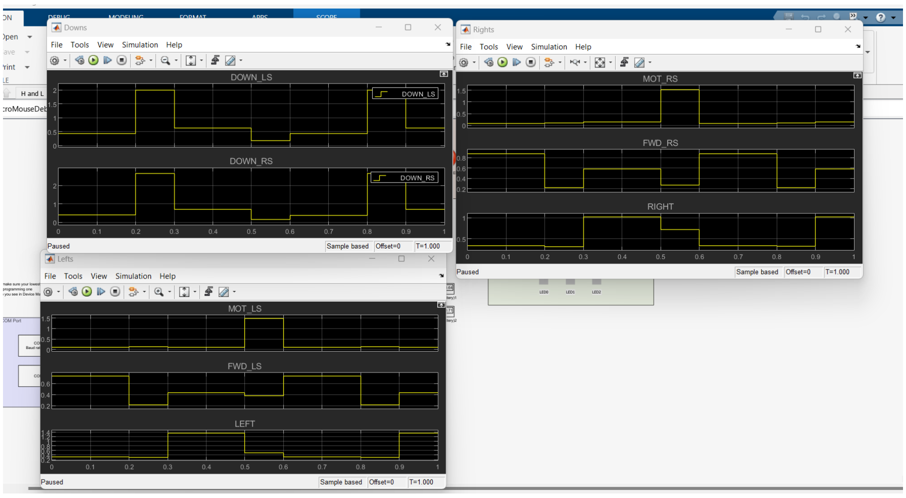


Figure A.9: IR LED Pattern

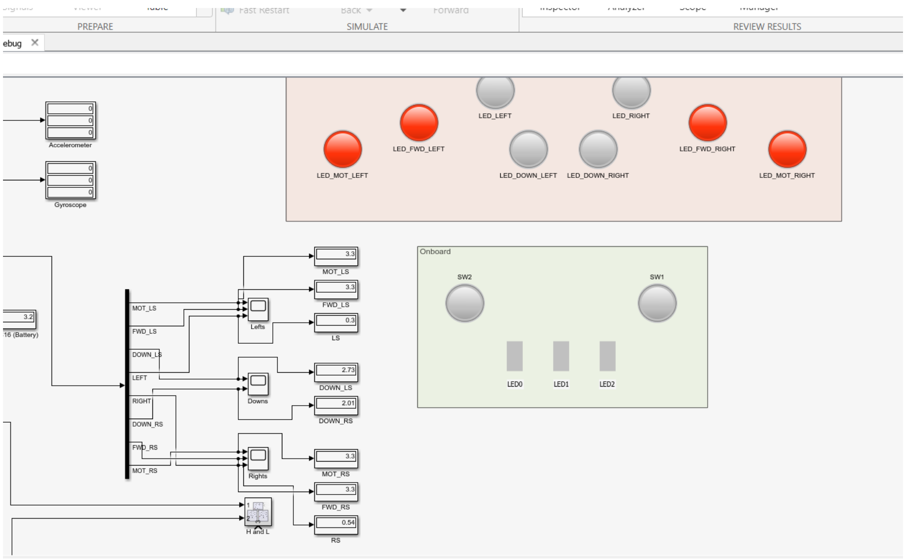


Figure A.10: Testing on black line present.