

# Milestone 3 &4

(Navigation and Optimization)



**Prepared by:** Manare Rammutla & Bohlale Thulare

**GROUP 16**

**Prepared for:** EEE3099/97F Department of Electrical Engineering  
University of Cape Town

**Submission Date:** 22 October 2024

## Table of Contents

A. Executive Summary of the full project .....	4
B. Design and Implementation of Navigation .....	6
1. Milestone 3 Navigation Integration to Reduce Limitation Design .....	6
1.1. Internal Limitations and Solutions .....	6
1.2. External Limitations .....	11
2. Non-Obvious, Original Solutions for Navigation .....	12
C. Design and Implementation of Optimization .....	14
1. Optimization algorithms designs analysis .....	14
2. Implementation of Optimization .....	15
D. Conclusion and Recommendations .....	18
1. Conclusion .....	18
2. Recommendations for Improvement .....	19

## 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 work.
4. We have not allowed, and will not allow anyone, to copy my work with the intention of passing it off as their own work or part thereof.
5. The following indicates the division of work:
  - The sections and work in this report were evenly distributed.
  - The Stateflow code in MATLAB during testing and debugging was evenly distributed



---

Manare Rammutla



## A. Executive Summary of the full project.

### Micromouse Design Project Summary

This project focuses on developing an autonomous Micromouse capable of efficiently navigating a maze by integrating sensor systems, real-time control algorithms, and performance optimization techniques. The design and implementation of subsystems were aimed at achieving stable, precise, and adaptable movement to address complex challenges.

#### Key Objectives of Milestone 3: Navigation [with Milestone 1 and 2 Review]

Table 1: Highlights the key achievements of Milestone 3.

Key Achievements	Description and Achievement Status
Hardware Assembly (Milestone 1)	Soldering and assembling the Micromouse and flashing it with initialization code. Achieved during milestone 1
Accurate line following (Milestone 2)	The Proportional-Integral-Derivative (PID) controller was fine-tuned to ensure stable, responsive control, enabling the Micromouse to follow the maze's central path with minimal deviation. <ul style="list-style-type: none"><li>Achieved during milestone 2</li></ul>
Obstacle Detection (Milestone 2)	180° infrared sensor coverage allowed the Micromouse to detect and avoid obstacles, making real-time adjustments to prevent collisions. <ul style="list-style-type: none"><li>Achieved during milestone 2</li></ul>
Dynamic Adaptability (Milestone 2)	The system's ability to adjust sensor thresholds in real-time based on changing environmental conditions ensured consistent performance, even under fluctuating light and surface reflectivity. Achieved during milestone 2
Accurate Turning (Milestone 3)	Using the feedback from the wall detecting sensors to turn 180° at a dead-end or 90° at an opening in the wall <ul style="list-style-type: none"><li>Yet to be achieved in milestone 3</li></ul>
Autonomous Navigation (Milestone 3)	Enable the Micromouse to navigate a maze autonomously, and dynamically adjust its path based on sensor feedback. <ul style="list-style-type: none"><li>Yet to be achieved in milestone 3</li></ul>

#### Key Objectives of Milestone 4

Table 2: highlights the key achievements of Milestone 4.

Key Achievements	Description
Optimized Navigation	Establish an algorithm that prioritizes paths that minimise turns and maximise straight-line movement to reduce the time required to complete the maze. This algorithm will identify the shortest and most efficient route.
Energy Efficiency	The path selection algorithm, combined with adaptive speed

<b>Improvements:</b>	control using IMU data, improved navigation efficiency, reducing both energy consumption and the time required to complete the maze.
<b>Improved Noise Handling</b>	By employing noise filtering and data averaging techniques, the Micromouse significantly reduced sensor inaccuracies caused by environmental noise, leading to more reliable operation.

### Project Overview

Throughout the project, multiple subsystems were designed and integrated to enhance the Micromouse's ability to perform accurate, real-time navigation. The sensor subsystem, comprised of infrared sensors and the IMU, was responsible for detecting obstacles, calibrating distances, and providing real-time orientation feedback. Redundant sensors and data averaging techniques were applied to reduce noise and increase reliability. MATLAB's Stateflow allowed for dynamic threshold adjustments, ensuring sensors-maintained accuracy across different environmental conditions, such as fluctuating light and surface reflectivity.

The PID controller was optimized to maintain a steady path by continuously correcting deviations, reducing oscillations, and improving overall stability. Adaptive speed control, informed by IMU feedback, enabled the Micromouse to accelerate on straight paths and slow down for turns, optimizing energy usage while ensuring precise movement.

In terms of optimization, a path selection algorithm was developed to find the most efficient route through the maze. This reduced overall completion significantly. Error reduction was further achieved by refining the calibration process, improving sensor accuracy, and adjusting the PID settings for quick recovery from minor deviations.

## B. Design and Implementation of Navigation

### 1. Milestone 3 Navigation Integration to Reduce Limitation Design

In Milestone 2, the focus was on the integration and calibration of the sensor subsystem, which provided the foundation for the Micromouse's ability to detect obstacles and maintain a steady path. However, several challenges were encountered during this phase, particularly regarding sensor noise, inconsistent distance measurements due to varying environmental conditions (e.g., lighting and surface reflectivity), and difficulties in achieving stable line following. These issues affected the accuracy of the Micromouse's obstacle detection and line-holding capabilities, causing deviations in its trajectory.

Internal and external factors significantly affect the Micromouse's ability to navigate through the maze. Internal factors include the motor speed inconsistencies, sensor sensitivity, sensor interference, and battery depletion, which all influence how the mouse moves. Externally, ambient light affected the performance of the sensors, leading to unreliable readings in environments with high reflectivity or high levels of natural light. The changing environment also played a role. These limitations are crucial to understand the how the mouse adapts to different maze conditions.

Milestone 3, which focused on navigation addressed these challenges by implementing a more refined control system and dynamic adaptation mechanisms. Specifically, improvements were made through the following approaches:

#### 1.1. Internal Limitations and Solutions

##### Internal Limitation 1: Motor inconsistencies

The left wheel consistently moved faster than the right wheel, and required manual calibration to equalize speeds. This effected the line following as well as the precision of the Micromouse turning and maintaining the line-following.

*Table 3: Motor Inconsistencies Limitations and solutions.*

Solution	Description	Drawbacks/Benefits	Implementation status
<b>Manual Calibration for Motor Speed</b>	Adjusting the speed of the left wheel by setting it to a slightly lower percentage (3% lower) to compensate for faster speed. E.g., left wheel at 73% while the right wheel is at 77%	<b>Benefit:</b> Equalized wheel speeds, improving straight-line movement. Simple implementation requiring no additional hardware. <b>Drawbacks:</b> Requires continuous recalibration, especially with changes in motor or battery performance.	<b>Implemented</b>

The following figure illustrates a Stateflow MATLAB code that supports the implementations in the above table.

```

LineFollowing

entry:
front = false;
sideSensors = 0;
leftWheel = 74;
rightWheel = 77;
LED1 = 0;
LED2 = 0;
during:
errorDown = D_LS - D_RS;
error = (errorInitial - errorDown)*100/4095;
leftWheel = 75+int8(0.5*error);
rightWheel = 75-int8(0.5*error);
OI = (F_RS > FrontRight && F_LS > FrontLeft);
JunctionR = (M_RS < JunctionRS);
JunctionL = (M_LS < JunctionLS);
exit:
leftWheel = 0;
rightWheel = 0;

```

Figure 1: Line following code that highlights Manual Calibration of wheel speeds

## Internal Limitation 2: Turning Challenges and Time Delay Dependency and Battery Power Constraints

As the Micromouse is being used the battery beings to discharge is affects the speed of the motors. This in turn changes the accuracy of the turning as well as the speed in which the Micromouse can navigate the maze. Increasing the battery life requires a hardware upgrade (buying a new battery), adds weight to the Micromouse and it adds cost. Using time-delays to turn the mouse was not effective as the speeds of the motors heavily relied on the battery being charged. As the battery depleted the motors would decrease in speed making the turns inaccurate

Table 4: Turning Challenges and Time Delay Dependency and Battery Power Constraints Limitations and solutions.

Solution	Description	Drawbacks/Benefits	Implementation status
<b>Frequent Recharging and reducing motor and sensor activity</b>	Recharging the battery frequently and minimizing motor and sensor activity when not needed to conserve battery life.	<b>Benefit:</b> Restores functionality temporarily and extends battery life by reducing power <b>Drawbacks:</b> Time-consuming recharging process and reduced performance, impacting speed and navigation precision.	<b>Implemented</b>
<b>Fixed Time Delays</b>	Initially, fixed time delays were used for turning based on expected motor speed and battery levels.	<b>Drawback:</b> As the battery drained, motor speed decreased, causing inaccuracies in turning. <b>Drawback:</b> Required constant recalibration, which was time-consuming.	<b>Considered, not implemented</b>
<b>Speed-Sensitive Time Adjustment</b>	Time delays were manually adjusted to match real-time motor efficiency as the battery discharged.	<b>Benefit:</b> Improved turn accuracy by aligning delay times with current motor performance. <b>Drawback:</b> Required frequent recalibration, especially for longer	<b>Considered, not implemented</b>

		operations, and didn't fully solve battery-related speed fluctuations.	
<b>Gyroscope Calibration</b>	Fine-tuning the gyroscope to calculate the optimal turning angle dynamically, compensating for motor variability.	<b>Benefit:</b> Achieved more consistent and accurate 90° and 180° turns regardless of battery charge level. More robust in dynamic environments. <b>Drawback:</b> Required extensive adjustments, and performance still depended on battery levels, necessitating further work on battery consistency.	<b>Implemented</b>

The following figure illustrates a Stateflow MATLAB code that supports the implementations in the above table. Figure 2 shows the time delay code which was not implemented, while figure 3 and 4 shows the implemented code of the gyroscope.

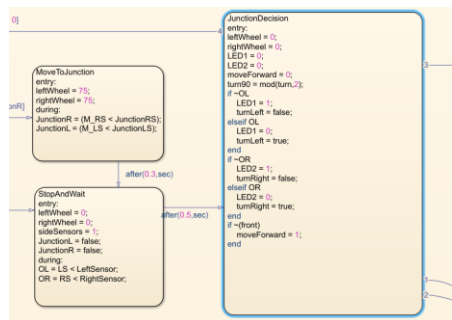


Figure 2: Initial Turning Using Time delay

```

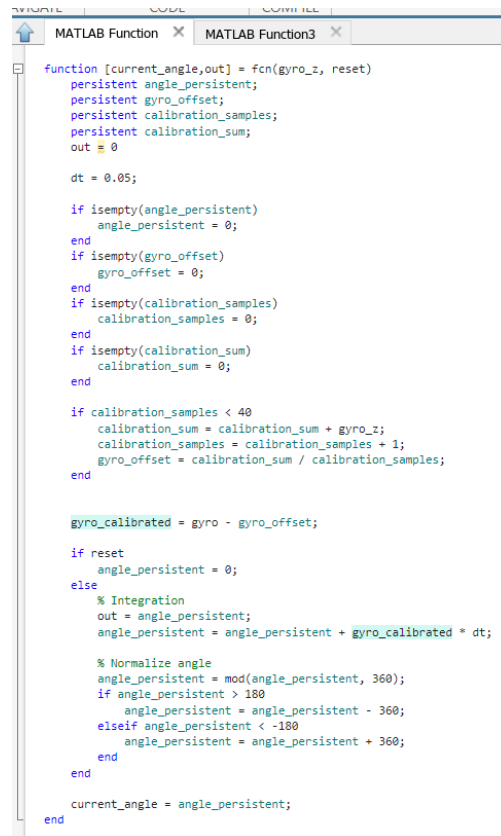
MATLAB Function2  x  MATLAB Function3  x
function [out, outIndicator] = fcn(in)
maxGyro = uint16((6.554e4) + 5);
outIndicator = uint16(0);

if in > (maxGyro - 360)
    out = uint16(maxGyro - in);
    outIndicator = uint16(1);
elseif (in == maxGyro)
    out = uint16(0);
    outIndicator = uint16(0);
else
    out = in;
end

```

Figure 3: Code for Turning Using the Gyroscope





```

function [current_angle,out] = fcn(gyro_z, reset)
persistent angle_persistent;
persistent gyro_offset;
persistent calibration_samples;
persistent calibration_sum;
out = 0;

dt = 0.05;

if isempty(angle_persistent)
    angle_persistent = 0;
end
if isempty(gyro_offset)
    gyro_offset = 0;
end
if isempty(calibration_samples)
    calibration_samples = 0;
end
if isempty(calibration_sum)
    calibration_sum = 0;
end

if calibration_samples < 40
    calibration_sum = calibration_sum + gyro_z;
    calibration_samples = calibration_samples + 1;
    gyro_offset = calibration_sum / calibration_samples;
end

gyro_calibrated = gyro - gyro_offset;

if reset
    angle_persistent = 0;
else
    % Integration
    out = angle_persistent;
    angle_persistent = angle_persistent + gyro_calibrated * dt;

    % Normalize angle
    angle_persistent = mod(angle_persistent, 360);
    if angle_persistent > 180
        angle_persistent = angle_persistent - 360;
    elseif angle_persistent < -180
        angle_persistent = angle_persistent + 360;
    end
end

current_angle = angle_persistent;
end

```

Figure 4: Code for Turning Using the Gyroscope continued

### Internal Limitation 3: Sensor interference

The infrared LED used for proximity sensing caused interference with nearby photodiodes, leading to inaccurate readings. This interference occurred when the LED's emitted light reflected off surfaces and was detected by adjacent photodiodes, overwhelming their input. As a result, the Micromouse struggled to differentiate between valid obstacle detection and internal interference, reducing sensor accuracy

Table 5: Sensor Interference Limitations and solutions

Alternative Solution	Description	Drawbacks/Benefits	Implementation status	
<b>Implemented Solution: PWM Adjustment</b>	Adjusting Pulse Width Modulation (PWM) to control the current sent to the sensors, reducing interference while maintaining accurate readings.	<b>Benefits:</b> Provides fine control over the sensor signal strength, allowing for reduced noise and interference. It is preventing cross-talk <b>Drawbacks:</b> Requires precise tuning to avoid underpowering or overpowering the sensors and adds complexity to sensor control logic.	Implemented	
<b>Black Tape</b>	Covering the photodiodes with	<b>Benefits:</b> Simple, effective solution.	Implemented	

	black tape to shield them from excess ambient light.	<b>Benefits:</b> Improved consistency in sensor readings and overall navigation reliability. <b>Drawbacks:</b> Requires manual installation and monitoring.	
--	--	--	--

The following figure shows the LEDs connected to the PWM

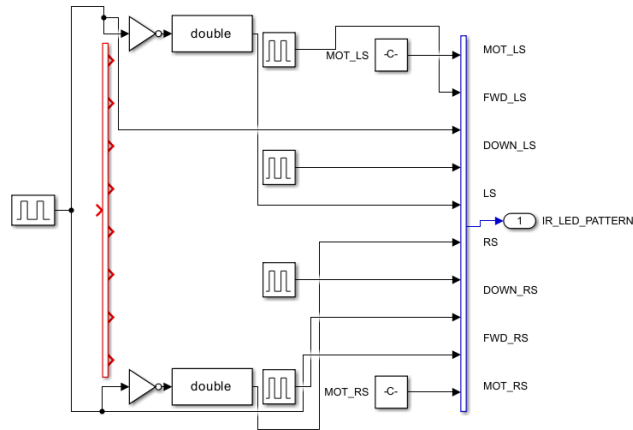


Figure 5: LED implementation of PWMs

#### Internal Limitation 4: Junction and Decision-Making

The Micromouse struggled with decision-making at junctions, often hesitating or making incorrect path selections due to unclear direction prioritization.

Table 6: Junction and Decision-Making Limitations and solutions

Solution	Description	Drawbacks/Benefits	Implementation status
<b>Timer-Based Decision System</b>	Using a timer to estimate when a junction is reached based on straight-line movement time.	<b>Drawback:</b> Unreliable due to varying motor speeds, leading to misidentification of junctions.	Considered, Not implemented
<b>Side Sensor Configuration</b>	Angling the side sensors downward to detect floor patterns and junction points, with halting for decision-making.	<b>Benefit:</b> Reliable detection and accurate decision-making at intersections. <b>Drawback:</b> Slightly slower navigation due to halting at junctions.	Implemented

The following figure shows the code for junction detection

```

JunctionDecision
entry:
leftWheel = 0;
rightWheel = 0;
LED1 = 0;
LED2 = 0;
moveForward = 0;
turn90 = mod(turn,2);
if ~OL
    LED1 = 1;
    turnLeft = false;
elseif OL
    LED1 = 0;
    turnLeft = true;
end
if ~OR
    LED2 = 1;
    turnRight = false;
elseif OR
    LED2 = 0;
    turnRight = true;
end
if ~(front)
    moveForward = 1;
end

```

Figure 6: Code for Junction detection

## 1.2. External Limitations

### External Limitation 1: Ambient light and sensor noise

Externally, ambient light affected the performance of the sensors, leading to unreliable readings in environments with high reflectivity or brightness

Table 7: Ambient Light and Sensor Noise Limitations and solutions

Solution	Description	Drawbacks/Benefits	Implementation status
<b>Software-Based Filtering</b>	Applying real-time filtering algorithms to smooth out noisy sensor data caused by ambient light.	<b>Drawbacks</b> :Ineffective in highly reflective environments. <b>Drawbacks</b> :Increased computational overhead, slowing down real-time processing.	Considered, Not implemented
<b>Hard-Coded Maximum Thresholds (Initial Solution)</b>	Using fixed maximum sensor thresholds to account for worst-case lighting conditions	<b>Benefits</b> : Simple and quick to implement and works well in controlled, consistent environments <b>Drawbacks</b> : Does not adapt to changes in lighting or environmental conditions so leads to inaccurate readings in dynamic conditions.	Not implemented (Replaced)
<b>Adjusting Sensor Sensitivity Settings</b>	Manually tuning the sensitivity of the infrared sensors to reduce their sensitivity and thus their response to ambient light by using the high current mode for the sensors responsible wall detection.	<b>Benefits</b> : Reduced interference from ambient light in some conditions, which in turn improves the line following	Implemented
<b>Black Tape</b>	Covering the photodiodes with black	<b>Benefits</b> :Simple, effective solution.	Implemented

	tape to shield them from excess ambient light.	<b>Benefits</b> :Improved consistency in sensor readings and overall navigation reliability.	
--	--	--	--

## External Limitation 2: Environmental Changes

Variations in maze environments, such as lighting and surface reflectivity, affected sensor accuracy and overall system performance, leading to navigation issues. If the thresholds were hard-coded into the system it would continuous have to be manual calculated and updated as the sun set, clouds passed, during the day or at night or if a shadow fell onto the maze.

*Table 8: Environmental Changes Limitations and solutions*

Alternative Solution	Description	Drawbacks/Benefits	Implementation status
<b>Dynamic Sensor Calibration</b>	It is dynamically calibrating sensor thresholds at the start of each maze run to adjust for changing lighting and environmental conditions.	<b>Benefit:</b> Adaptable to varying maze and lighting conditions, improving sensor performance in real-time <b>Drawback:</b> Slightly increases setup time before each maze run	<b>Implemented</b>

Through these integrations, the Micromouse was able to navigate the maze more reliably by addressing sensor noise, enhancing decision-making at junctions, managing power-related motor inconsistencies, and fine-tuning turning accuracy. While certain limitations persisted, these solutions significantly improved the system's navigation capabilities and adaptability to dynamic conditions within the maze. The system adheres to common electrical and mechanical standards for motor control and sensor integration. The navigation solution was designed to be adaptable to any maze configuration by relying on sensor data and feedback loops rather than hard-coded movements. This allows the Micromouse to adjust to various maze layouts and conditions, ensuring flexibility across different environments.

## 2. Non-Obvious, Original Solutions for Navigation

The Micromouse's navigation system required non-trivial solutions to meet the complex challenges posed by the maze. One of the key original solutions involved bending the side sensors down to detect lines at junctions. This innovative use of the sensors enabled the Micromouse to more accurately detect turns at junctions, improving decision-making and reducing errors.

Furthermore, the use of the gyroscope to control turning was another original approach. Gyroscopes measure the rate of rotation and angular velocity, allowing for precise control over rotational movements. By using gyroscope data instead of relying solely on motor timing, the Micromouse could execute accurate 90 ° and 180 ° turns, even under varying

battery conditions. This solution addressed the issue of inconsistent motor performance caused by battery voltage changes, providing a robust and reliable way to handle turns.

Another example of originality was the method used to manage sensor interference. By implementing a PWM-based system, sensors not in use were turned off to prevent cross-talk and interference between them. This improved the accuracy of the active sensors, ensuring more reliable obstacle detection and wall-following.

## C. Design and Implementation of Optimization

### 1. Optimization algorithms designs analysis

The following table illustrates the methodologies and alternatives that are to be implemented on milestone 3 to improve requirements set.

Table 9: Optimization designs

Algorithm	Description	Benefits/Drawbacks	Implementation Status
<b>Dijkstra's Algorithm</b>	Finds the shortest path by exploring the node with the lowest cumulative weight (cost) at each step	<b>Benefits:</b> Guarantees the shortest path in weighted mazes and it works well for mazes with varying traversal costs. <b>Drawbacks:</b> It is Computationally expensive and it overkill for simple or unweighted mazes.	Not Implemented
<b>Breadth First Search(BFS)</b>	Explores all paths one step at a time, ensuring all nodes are visited level by level.	<b>Benefits:</b> Guarantees finding the shortest path in an unweighted maze and Systematic exploration. <b>Drawbacks:</b> High memory usage (stores all nodes). It is Slower due to broad exploration before going deep.	Not Implemented
<b>First Right following</b>	The Micromouse always turns right at each junction until it finds the exit or a dead end, then backtracks	<b>Benefits:</b> Simple to implement and it does not require much computation or memory. <b>Drawbacks:</b> Not optimal; can get stuck in loops. It fails in complex mazes with multiple branches, and it does not guarantee shortest path	Not Implemented (drawbacks outweighs simplicity)
<b>Depth First Search (DFS)</b>	Explores each path as far as possible before backtracking.	<b>Benefits:</b> Easy to implement. Efficient in memory usage and systematically explored paths <b>Drawbacks:</b> Can get trapped in long, deep branches (if not backtracked efficiently). Not optimal for shortest paths.	Implemented

The algorithms that are being considered for maze navigation are summarised in the table, with an emphasis on their advantages and disadvantages. Breadth First Search (BFS) and Dijkstra's Algorithm were not implemented due to their high memory consumption and computational complexity. Although these algorithms ensure the shortest path in mazes, their complexity rendered them unsuitable for this project. In the same vein, the First Right Following algorithm was not selected due to its tendency to become entangled in cycles and

its inability to efficiently navigate complex mazes, despite its simplicity of implementation and minimal computation.

However, the initiative implemented Depth First Search (DFS) as a result of its efficient memory usage and simplicity. DFS is highly systematic in its exploration of the maze, as it investigates each path to the maximum extent possible before retracing its steps. Although it does not guarantee the shortest path and can occasionally result in deep branch traps, its minimal memory requirements and ease of implementation made it the optimal choice. The potential drawbacks of DFS were outweighed by the project's requirements for efficient memory utilisation and comprehensive path exploration, resulting in its successful application in the maze.

## 2. Implementation of Optimization

Optimisation was crucial in the Micromouse's navigation system to guarantee optimal performance in intricate maze environments. Key adaptations were developed to address the specific challenges presented in the project, while well-established algorithms and techniques were leveraged. The primary navigation method was the Depth First Search (DFS) algorithm, as illustrated in the accompanying diagram, which was deployed among these solutions. This section defines the primary optimisations that were implemented, such as the calibration of motors, the adjustment of sensors, and the implementation of DFS.

### 1. Depth First Search (DFS) Optimization

The DFS algorithm was selected for its systematic exploration of each path until it reaches an end, and then it backtracks to explore other branches. The execution of DFS is visually represented in the flowchart provided, which commences with the initialisation of the stack with the start node and the designation of nodes as visited. The algorithm continues to investigate neighbouring nodes, adding them to the array and determining whether the objective has been achieved or if there are no remaining paths.

The figure above visually illustrates how DFS operates through:

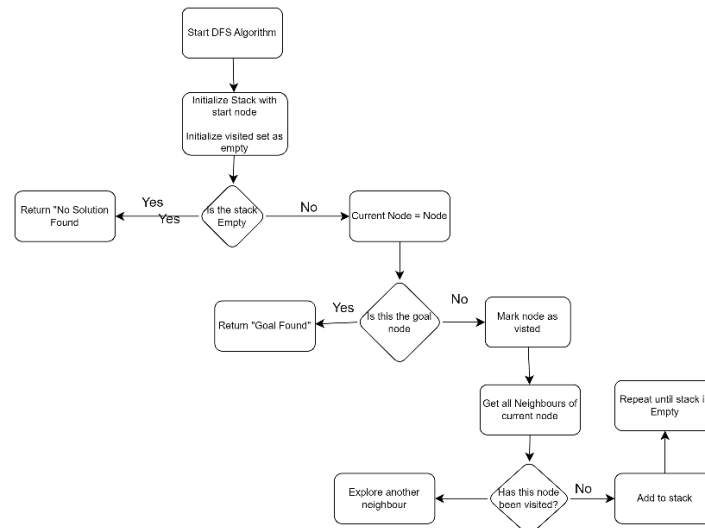


Figure 7: The pseudocode flowchart of the Depth First Search algorithm

- **Initialization:** The start node is used to initialise an array, and a set of visited nodes is initialised as empty.
- **Exploration:** The current node is verified to determine if it is the target at each stage. The algorithm designates the node as visited, explores its neighbours, and repeats the process until all possibilities are exhausted or the goal is found if not.
- **Backtracking:** In the event that the present path results in a dead end, the algorithm retraces its steps to investigate alternative paths from previous nodes.

Nevertheless, as the flowchart illustrates, DFS is not intended to identify the shortest path as it maps and may occasionally become entangled in lengthy branches. This disadvantage was alleviated by restricting the investigation of deep branches and by effectively backtracking when required, allowing the shortest path to be revealed after mapping and thus the final demo is efficient.

## 2. Motor Speed calibration for precision

The limited battery life presented a substantial obstacle to attaining energy efficiency, particularly in spite of the motors' high-power requirements. An adaptive speed control system was created to address this issue by utilising real-time IMU data. This system enables the Micromouse to intelligently accelerate on straight sections and reduce speed when executing turns. This optimisation facilitated the conservation of battery life by adjusting the speed in accordance with the present power output of the motors, thereby achieving a balance between energy consumption and performance. Nevertheless, battery constraints continued to be a constraint during extended duties, despite these enhancements. The identification of future potential enhancements, such as the introduction of energy-efficient motors or higher-capacity batteries, was the result of an analysis of energy efficiency. This was done to extend the operational time without compromising navigation accuracy.

The Micromouse only has to stop and make decisions at each junction when it is mapping the maze. When the Micromouse has the shortest part sorted and attempts to solve the maze as fast as possible it does not need to stop at the junctions only needs to count the nodes to reach its goal (the centre block)



### **3. Sensor Calibration and Adaptation**

Numerous optimisation strategies were implemented to mitigate sensor interference. At the commencement of each maze run, dynamic sensor calibration was implemented to ensure that sensor thresholds were adjusted to reflect the current environmental conditions, including variable illumination. This method offered adaptability in contrast to the hard-coding of sensor thresholds, which may not function properly in dynamic environments.

Furthermore, the ability to average sensor measurements enabled the identification of obstacles with greater precision and dependability. This enhanced the overall accuracy of navigation by mitigating the effects of sensor noise and inconsistencies.

### **4. Gyroscope -Based Turning**

The integration of gyroscopic data was employed to overcome the challenge of achieving precise turns in the maze, thereby avoiding the ineffective use of time-delays that were caused by motor speed variations resulting from battery depletion. Real-time adjustments at decision nodes were enabled by gyroscope-based turning, which guaranteed the consistent execution of 90° and 180° turns. Nevertheless, the precision of the turn was still influenced by the variability of motor performance, which was primarily due to battery discharge. To address this issue, the gyroscope was calibrated iteratively to ensure that the turning angles were consistent with the motor speeds in the presence of variable power levels. Original solutions, such as power-based time adjustments, were devised by examining the impact of battery health on both time delays and turn accuracy. The performance of the system was considerably enhanced by these modifications. However, future optimisations could concentrate on improving the gyroscope's responsiveness and reducing the system's dependence on battery-dependent delays to ensure even more consistent and precise turns in a variety of conditions.

## D. Conclusion and Recommendations

### 1. Conclusion

Performance Optimisation was not completely realised during the demonstration, despite the fact that the Micromouse project successfully achieved several key objectives related to autonomous navigation, path following, and obstacle detection. The Micromouse prematurely turned at the junction, rather than proceeding straight until it encountered a wall. The intended design was for the mouse to proceed forward and only turn at the last available opening, but this behaviour deviated from that. Additionally, the system's capacity to optimise subsequent decisions based on previously visited paths was restricted by its failure to accurately save the maze as it navigated.

Below is a summary of the objectives and how they were met, except for the Performance Optimization:

*Table 10: Results and Performances of Objectives*

Objectives	Results/Performance
<b>Autonomous Navigation</b>	The Micromouse navigated autonomously, using DFS to explore paths, but decision-making at junctions was not fully optimized.
<b>Precise Path Following</b>	The PID controller enabled stable and accurate path following with minimal deviations.
<b>Obstacle Detection and Avoidance</b>	Real-time obstacle detection was successfully achieved using infrared sensors, allowing the Micromouse to avoid walls.
<b>Adaptive Calibration and Speed Control</b>	Dynamic adjustments were made to sensor thresholds and motor speeds based on environmental conditions.
<b>Performance Optimization</b>	<b>Not fully achieved:</b> The Micromouse turned prematurely at junctions, failed to save the maze layout, and did not optimize decisions at intersections.

#### Possible reasons for Suboptimal Performance

- i. **The improper handling of Junctions:** The logic employed to manage junctions may not have adequately considered the necessity of proceeding until one encounters a wall. This could have been the result of an excessively sensitive sensor response to junctions, which led the Micromouse to prematurely regard the junction as a turn point.
- ii. **Limitations of the DFS Algorithm:** Although DFS is effective for systematic investigation, its inability to comprehend the maze's overall layout (due to the maze's non-saved state) resulted in inefficient decisions at junctions. The premature turning behaviour may have been the result of DFS's depth-first approach, which neglects the broader context.
- iii. **Sensor Inaccuracy or Noise:** The Micromouse's misinterpretation of the environment, particularly at junctions, may have been influenced by sensor inaccuracies or noise. This could account for the fact that it turned prematurely rather than continuing forward.

## 2. Recommendations for Improvement

The following recommendations are suggested to address the identified performance issues and improve the overall efficacy of the Micromouse. These enhancements will concentrate on the optimisation of junction management, maze mapping, and sensor calibration to guarantee more precise and consistent navigation in future implementations.

1. **Junction Handling Logic:** Conduct a review and modification of the junction detection logic to guarantee that the Micromouse proceeds straight through intersections until it reaches a wall, at which point it will turn at the last available opening. Implement additional tests at junctions to guarantee that turns are only executed when they are required.
2. **Enhance Maze Mapping Capabilities:** Incorporate a maze-saving mechanism that is efficient in storing visited paths and decisions. The Micromouse can make more informed decisions at each junction and prevent revisiting previously explored areas by saving the maze, thereby improving performance.
3. **Improve Sensor Calibration:** Recalibrate the sensors to minimise false positives and noise at junctions. This will guarantee that the Micromouse interprets junctions accurately and does not prematurely turn.
4. **Test in More Dynamic Environments:** Perform additional tests in dynamic or variegated maze environments with a variety of junctions and obstacles to enhance comprehension of the current system's ability to manage complex situations and to refine the logic for handling junctions.