Final Report

Prepared by:
Liyana Singh - SNGLIY001
Luke van der Walt - VWLLUK003

Report Contributions:
Liyana Singh
chapter 1 : 30% chapter 2 : 90% chapter 3 : 50% chapter 4 : 70% chapter 5 : 10%
Luke van der Walt
chapter 1 : 70% chapter 2 : 10% chapter 3 : 50% chapter 4 : 30% chapter 5 : 90%

Design Contributions:
Liyana Singh - Maze Solving and Optimization
Luke van der Walt - Movement and Navigation

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

October 21, 2024

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I 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 my own work.
4. I 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.

October 21, 2024

Liyana Singh — Date

October 21, 2024

Luke van der Walt — Date

Contents

Chapter 1

Executive Summary

1.1 Problem Description

This report is based on Milestone 3 and Milestone 4 for the micro-mouse project and focuses on the development of the navigation through the maze and optimizing the route taken through the maze. The micro-mouse project was assigned by the EEE3099S course within the EBE faculty at the University of Cape Town and encompasses the software development for a micro-mouse robot. The project has been split into four milestones, which are Milestone 1: Status Report, Milestone 2: Localization, Milestone 3: Navigation and Milestone 4: Optimisation.

A micro-mouse is a wheeled robot designed to navigate through a maze using sensors and control algorithms. The goal is for the robot to autonomously find the most efficient route to the center of a maze. In this project, the maze will be constructed physically using wooden walls and black tape marking the floor, and the robot will use infrared sensors to detect the walls and floor lines. It has an STM32 microcontroller onboard that it will use to make decisions based on what it senses, and to map out the maze in real time to solve it.

1.2 Scope and Limitations

The scope of Milestone 3 involves developing the navigation logic for the micro-mouse to move through the maze. This includes the control of the sensors and motors. The scope of Milestone 4 involves developing the optimisation logic to minimize errors made while navigating through the maze and minimize the time taken to solve the maze. The scope does not include Milestone 2 which focused on developing the sensing functions and basic motor control. Milestone 2 included the initialization and calibration needed for the micro-mouse to have surrounding awareness of its environment and this was successfully achieved. The micro-mouse's functional state going into Milestones 3 and 4 was that is could calibrate itself and use these calibration values to track a black line and detect walls around it.

The limitations are listed in Table 1.1. Attempts will be made to mitigate the effects of these limitations, which will be discussed within the design choices in chapter 3.

Table 1.1: Limitation Table

Limitation ID	Limitation
L1	The IR emitting diodes emit a weak light.
L2	The IR photo-transistors are heavily affected by ambient light from the environment
L3	The IR photo-transistors are heavily affected by the IR light produced by the other IR emitting diodes.
L4	The IR emitting diodes and IR photo-transistors have been placed on the PCB and cannot be moved.
L5	The motors have a maximum and minimum speed.
L6	The development environment available is Matlab which has a absolute maximum sampling speed of 0.01s or 0.05s with the ability to debug.

1.3 Achievements

The achievements matrix in Table 1.2 outlines all the requirements for Milestones 2, 3, and 4. Each design element has a colour-coded indicator to signify its status - whether it has been achieved, remains outstanding, or is not applicable to the respective milestone requirement.

Colour Key:	Achieved	Outstanding	Not Applicable

Table 1.2: Achievements

	IR Emitting Diodes	Motors	Calibration	Movement (Logic)	User LEDs	SMD
Calibrate your mouse						
Perform initialization routines						
Identify and hold a line						
Surrounding awareness						
Moves, decide and make turns						
Optimize the path through the maze						
Minimize errors and maximize speed to complete the maze						

Chapter 2

Requirements Analysis

2.1 Requirements

The micro-mouse must meet the following requirements outlined in Table 2.1.

Table 2.1: Requirements

Requirement ID	Requirement
R1	Autonomously make decisions in the maze
R2	Precise turns to navigate the maze effectively
R3	Optimization

2.2 Specifications

The requirements in Table 2.1 are broken down into their respectful specifications in Table 2.2.

Table 2.2: Specification

Specification ID	Specification
S1	The micro-mouse makes a decision to enter/leave each pixel.
S2	The micro-mouse makes a 90° left turn.
S3	The micro-mouse makes a 90° right turn.
S4	The micro-mouse makes a 180° U-turn.
S5	Minimize the distance of the path taken.
S6	Maximise speed of the micro-mouse.

2.3 Testing Procedure

Table 2.3 is a summary of the acceptance testing procedures which will be fully explained in chapter 4.

Table 2.3: Summary of Acceptance Testing Procedures

Test ID	Test
AT1	Compare the micro-mouse's decision in the maze to the expected decision based on the optimization algorithm at each pizel.
AT2	Place a wall to the right, in front and behind the micro-mouse. Observe the micro-mouse.
AT3	Place a wall to the left, in front and behind the micro-mouse. Observe the micro-mouse.
AT4	Place a wall to the left, in front and to the right of the micro-mouse. Observe the micro-mouse.
AT5	Create a simulation of the maze and calculate the shortest path(s) to the destination. Place micro-mouse in the physical maze and observe the path taken.
AT6	Time the micro-mouse and compare to previous times taken to solve the maze.

2.4 Traceability Analysis

Table 2.4 shows how the requirements, specifications and testing procedures all link.

Table 2.4: Requirements Traceability

#	Requirements	Specifications	Acceptance Tests
1	R1	S1	AT1
2	R2	S2 S3 S4	AT2 AT3 AT4
3	R3	S5 S6	AT5 AT6

Chapter 3

Design

3.1 Navigation

3.1.1 Decision Making

Specifications

The micro-mouse must be able to decide when to turn left, right, go straight or turn around according to S1, S2, S3 and S4 as detailed in Table 2.2. For the micro-mouse to make a turning decision, it must first determine the appropriate timing for the decision by knowing its relative position in the maze so that it can be aware of any surrounding walls that prevent it from turning.

Possible Design Solutions

One method to know its relative position would be to measure the distance travelled by measuring the wheel rotations and comparing this distance to the fixed length of a maze pixel. This would enable the mouse to know what block it was in and where in that block it was situated. This would involve painting reflective stripes on the wheels to count the number of rotations. This method would rely on the accuracy of the measured position, which is susceptible to change over longer distances if the calculated values are slightly off. Another method would be to record every time it reaches a junction (line-crossing) in the maze. Since the size of the maze and the starting position are known, the crossings give an accurate show of the relative position compared to the destination pixel. This method is simpler than recording the total covered distance, however, if the micro-mouse misses a single crossing it will become lost. Additionally, this method also limits the micro-mouse to moving solely in the middle of the blocks constantly, meaning that it would have to stop at a junction to turn, unlike the first method where it could predict gradual turns, making it faster. However, because of the simplicity and limited error build-up over time, the junction position method was chosen over measuring the distance.

Final Design

The micro-mouse follows a black line on the floor using the bottom-facing infrared receivers. When the mouse is on the black line, the receivers' values are lower than their value compared to the normal wood floor surface. If the micro-mouse reaches a junction, the received infrared light value drops to a distinctive low value. When this value is sensed, the micro-mouse knows it has reached a junction and first takes readings of its sensor values of the left, right and front-facing infrared receivers. It then calls a maze solving function, as detailed in subsection 3.2.1, that takes in these receiver values, updates the micro-mouse's current position, and returns a direction variable to tell it which way to turn.

3.1.2 Turning

Specifications

The micro-mouse must be able to make accurate 90° and 180° turns to maintain a correct heading according to S2, S3 and S4 as detailed in Table 2.2. These turns must be consistent and allow it to continue navigating the maze without interruption.

Possible Design Solutions

One method to turn a specific amount is to turn the micro-mouse at a known speed for a specific amount of time. Using a fixed turning speed means the time taken for either 90° or 180° turns can be found and the micro-mouse can be turned for a specified amount of time to achieve these rotations. This method is simple and relies on the consistency of the time and motor speed used, which are fairly consistent for the micro-mouse, however, it also relies on the angle at which the micro-mouse approaches a junction, which is determined by the accuracy of the line-following. If the angle of approach is too far from the expected, the micro-mouse will turn for either too long or too short and lose track of the solid line. A similar method to this is for the micro-mouse to reach a junction and start turning and only stop when it sees the next line. This method is simple and works well provided the next line the micro-mouse sees is the desired one, otherwise it finds the incorrect line and is lost. Another method is to use the onboard gyroscope to keep track of the angle the micro-mouse has turned by integrating the value recorded by the gyroscope. This method can produce very accurate turns but is more complex and would take longer to implement than the timing method. Since the line following implemented in Milestone 2 was fairly accurate, the less accurate but simpler timing method was used in conjunction with the line-checking method as the chosen solution.

Final Design

When the micro-mouse reaches a junction, it determines whether or not it will turn. If it decides to turn, it will turn on its motors in opposite directions at the same duty cycles which will start turning the micro-mouse about its center. This will continue for a specified calibrated time. Since this method is susceptible to error if the initial starting angle is not what the micro-mouse anticipated, an additional check is added to the micro-mouse while it is turning to detect whether or not a line is detected. If a line is detected, it stops turning for the set amount of time and assumes that it has found the line. This checking only begins after a specified time to ensure that the micro-mouse doesn't accidentally detect the forward-facing line if its angle of approach is very off in the opposite direction to the turning direction. An example of the code used to turn a left corner is shown in Figure 3.1 below, the same approach was used to turn right and around.

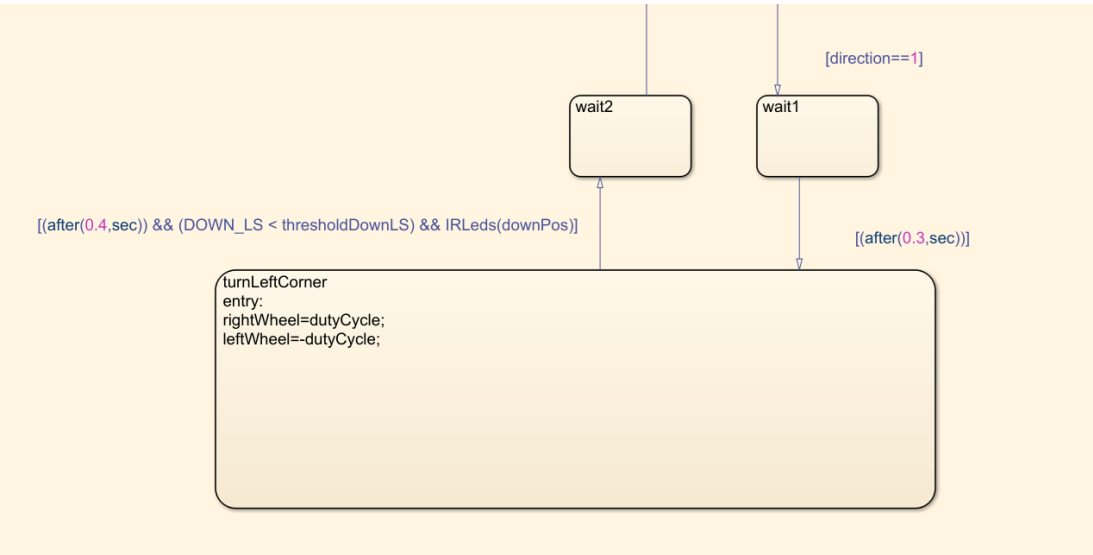

Figure 3.1: The Code Used to Turn Left

3.2 Optimization

3.2.1 Maze Solving Algorithm

Specifications

To satisfy the specification S5, minimize the distance of the path taken as shown in Table 2.2, a maze solving algorithm must be implemented. The micro-mouse receives data from the sensors and, using the algorithm, accurately determines the shortest distance the micro-mouse should travel to reach the final destination.

Possible Design Solutions

The left wall tracking and floodfill algorithms were considered. Left wall tracking is a simple approach where the micro-mouse always follows the wall to its left. It is easy to implement but it may not always find the shortest path and can get stuck in a loop in certain maze configurations. In contrast, the floodfill algorithm is more complex. It assigns values to each pixel based on its distance from the destination. These values represent how many steps the pixel is from the destination factoring in any walls in the maze. The micro-mouse then moves from its current pixel to the neighboring pixel with the lower value. This method guarantees the optimal path but requires more processing power and memory. The floodfill algorithm was chosen to ensure reliability and adaptability to any maze.

Final Design

The final design was implemented using the Maze, Update Vertical Walls, Update Horizontal Walls, Update Distance and Turn Decision Functions. The floodfill algorithm is the main component of the Update Distance Function.

At each junction, the Maze Function is called. The purpose of this function is to output the direction the micro-mouse must move (left, forward, right or backward). The Maze Function must first update the walls detected by the micro-mouse. Then, uses the walls detected to update the distance each pixel is from the destination pixel. Finally, the function should determine which direction the micro-mouse should move in to find the destination in the shortest path.

The Maze Function is outlined in Figure 3.2. Four other functions are called in the Maze Function which perform certain tasks. These functions, namely, Update Vertical Walls Function, Update Distance Function and Turn Decision Function will be explained in conjunction with their flow charts in Figure 3.3, Figure 3.4 and Figure 3.5, respectfully.

The Update Vertical Wall Function is outlined in Figure 3.3. The Update Horizontal Wall Function operates similarly. The vertical wall array and horizontal wall array are imperative to the Update Distance Function.

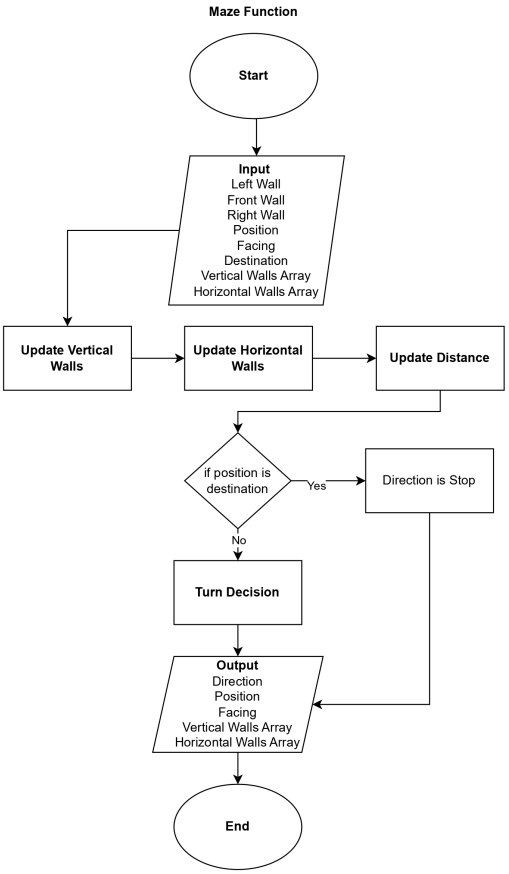

Figure 3.2: Maze Function Flow Chart

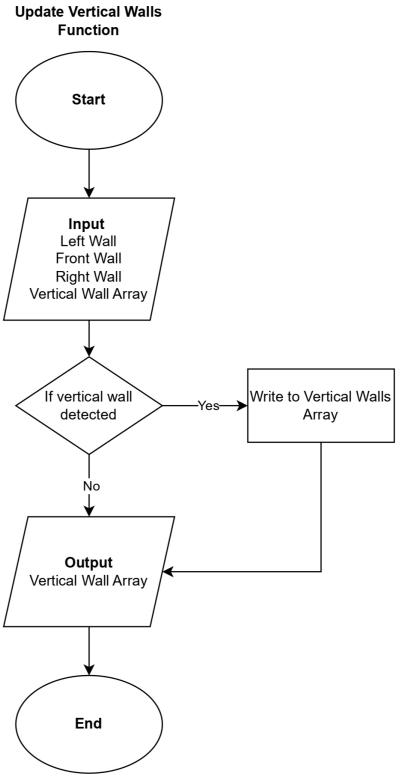

Figure 3.3: Update Vertical Walls Function Flow Chart

The Update Distance Function requires an end destination and two input arrays. Firstly, the horizontal walls array stores the horizontal walls of the maze and is updated when the micro-mouse reaches a junction. Secondly, the vertical walls array stores the vertical walls of the maze and is updated when the micro-mouse reaches a junction. The function outputs the distance array which is used in the Turn Decision Function to ensure the micro-mouse follows the shortest path. The distance array stores the number of steps each pixel is from the destination, taking into account any walls in the maze. The Update Distance Function Flow Chart in Figure 3.4 illustrates how the floodfill algorithm was used to determine the distance array.

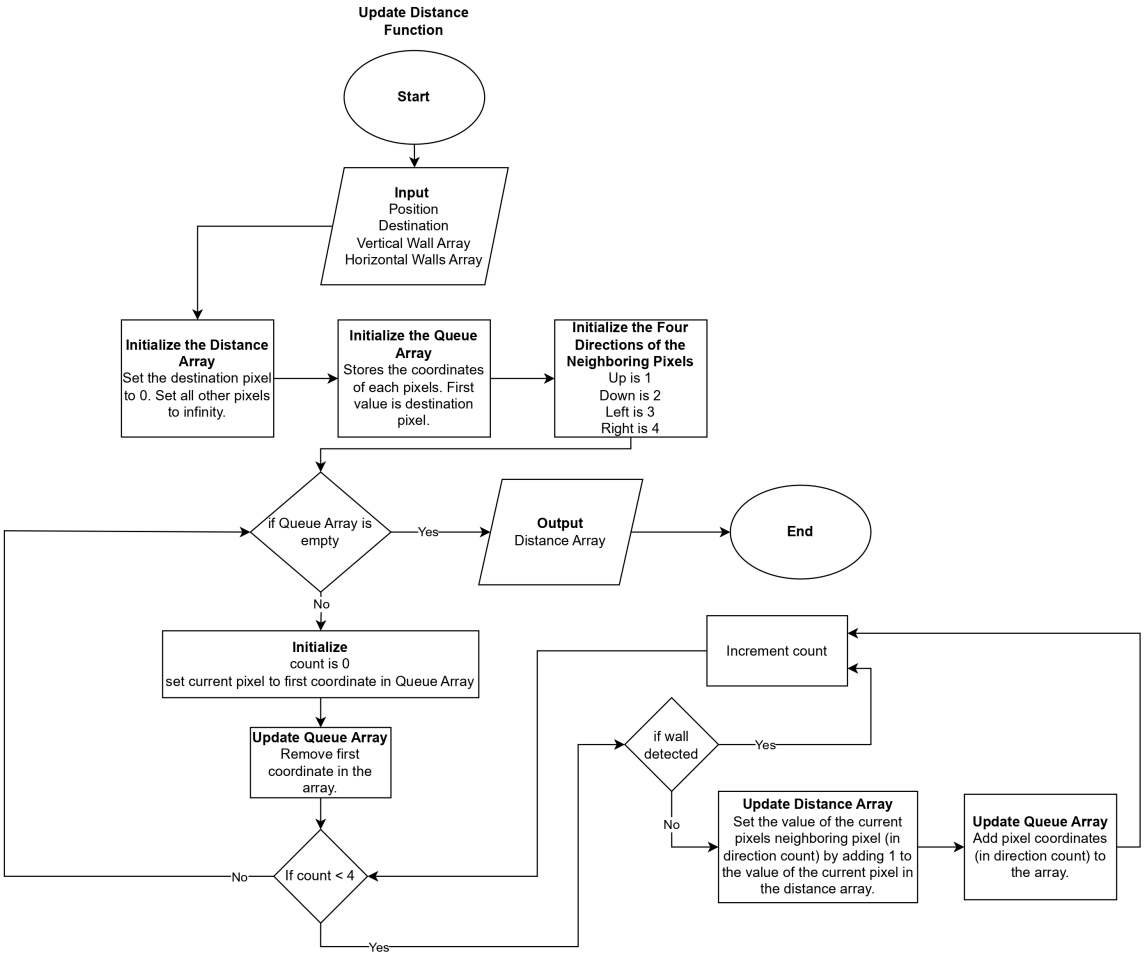

Figure 3.4: Update Distance Function Flow Chart

The Turn Decision Function uses the distance array which describes the shortest path from any pixel to the destination in order to output the correct direction the micro-mouse must turn. It also outputs the new position and the new direction the micro-mouse is facing to be used when the function is called again. This process is conveyed in Figure 3.5.

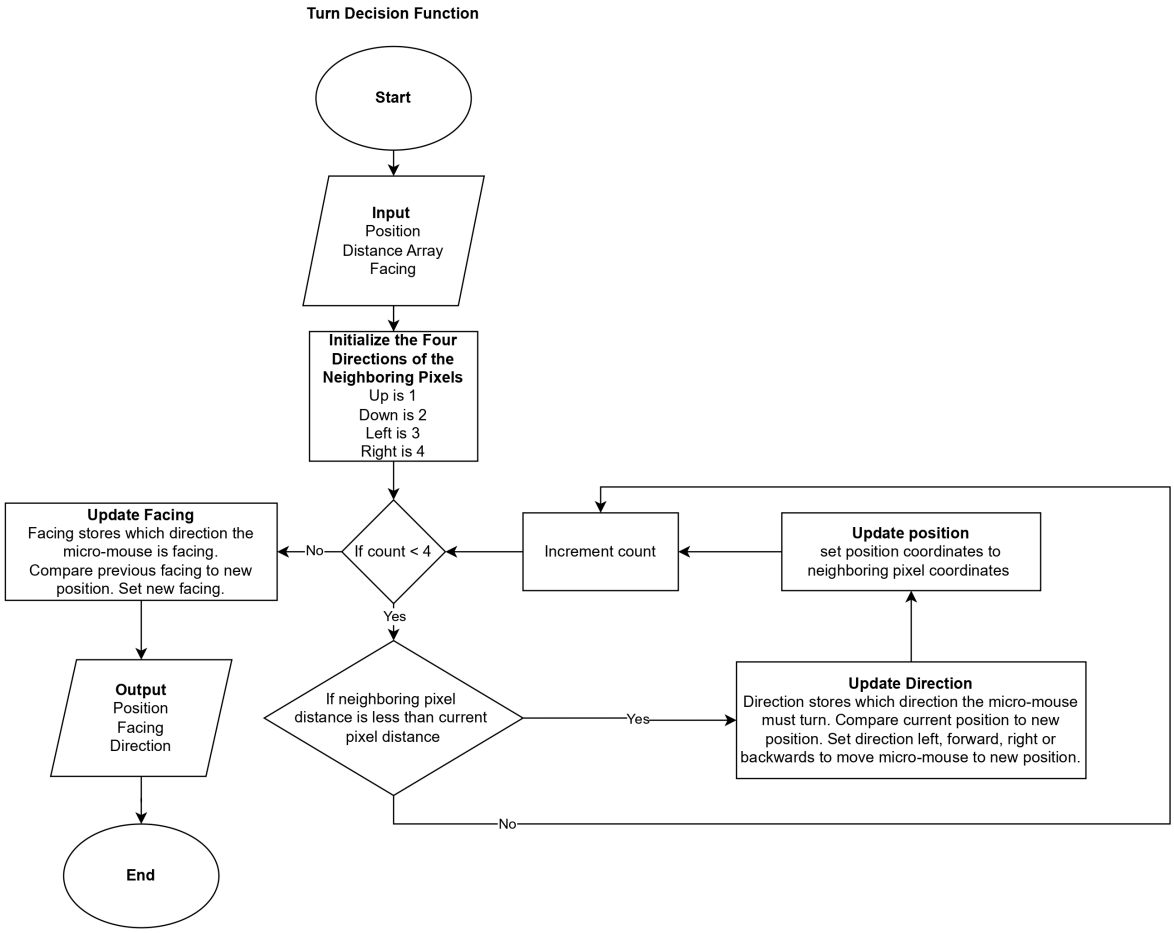

Figure 3.5: Turn Decision Function Flow Chart

The micro-mouse initially assumes the maze only has the outer walls as seen in Figure 3.6. The micro-mouse will explore the maze twice. Firstly, to reach the destination, then to return to the starting point. After the micro-mouse has explored the maze, the Maze Function, outlined in Figure 3.2, will have been called several times in different pixels across the entire maze. The micro-mouse will begin from the starting position for the third and final run. The micro-mouse will use the information gathered as seen in Figure 3.7 to optimise it's path to the destination. The starting and end points are highlighted in blue.

6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1	0	1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6

Figure 3.6: Maze with Outer Walls and Distance Values

14	13	12	13	14	15	14
15	12	11	10	11	12	13
14	13	10	9	8	7	6
15	12	11	0	1	4	5
16	17	12	1	2	3	4
15	16	11	10	11	6	5
14	13	12	9	8	7	6

Figure 3.7: Maze with All Walls and Distance Values

3.2.2 Speed

Specifications

The micro-mouse must solve the maze as fast as possible according to S6 as detailed in Table 2.2. To satisfy this specification, the mouse must run at the fastest speed while maintaining a high enough degree of accuracy to complete the maze successfully.

Possible Design Solutions

The slower the micro-mouse runs the more accurate it is, since it has more time to correct itself if it is unaligned. The minimum duty cycle of the micro-mouse's was 72% since anything lower would not allow it to move. Ideally, the micro-mouse would run at 100% duty cycle constantly throughout the maze to achieve the fastest time. Running it at these speeds greatly increases the chance of a fault occurring especially at a junction. Conversely, if the micro-mouse were to run at 72% duty cycle constantly, it would have a much lower chance of veering off the path at the cost of being much slower. One method to achieve good speed with good accuracy would be to run the micro-mouse at a variable speed depending on how well it is tracking. If the micro-mouse detects that it is slightly off, it will slow down to readjust and then speed up again once it is on track. This method would work well at maintaining the accuracy and speed of the micro-mouse, but is complicated since the micro-mouse needs to be able to quantify how inaccurate it is at any moment. Another simpler method would be to only change the speed of the mouse when it reaches a junction since this is the point where errors are most likely to occur and the point where it is most crucial to be accurate. The mouse could just have two set speeds, a faster one for line following, and a slower one for turning. This method may not be as fast as using a variable speed but would be much simpler to implement and still be reasonably fast with good accuracy which is why it was the chosen solution.

Final Design

The micro-mouse uses two duty cycle values as its speeds in the maze. The slower is set to 72%, which is the slowest the mouse can operate at, and the faster is 90%. When the micro-mouse is tracking a straight line, its speed is set to the faster option. The moment it detects a junction the speed is set to the slower option. Slowing the micro-mouse down at the junction guarantees it has enough time to process any decisions and additionally reduces the chance of its inertia causing it to deviate from the path. This faster speed used while tracking the line doesn't produce more errors because the line following used is fairly robust and can handle faster speeds. This method therefore provides high accuracy at reasonable speeds.

3.3 Additional Error Mitigation

3.3.1 Ambient Light Conditions

The micro-mouse calibration was initially catered towards medium-brightness ambient light conditions. This resulted in it having inconsistent behaviour in very bright conditionals. To solve this the previous calibration technique developed in Milestone 2 was replaced with an improved solution that included a reading of the ambient light. By taking the difference of the infrared receiver values when the infrared emitters were on, to when they were off, the effect of the ambient light could be mitigated. This approach effectively just measured the contribution of the reflection of the infrared emitters, which made the wall and line detection more accurate in bright conditions.

Chapter 4

Results

4.1 Tests

Table 4.1 states the condition for the micro-mouse to pass each acceptance test.

Table 4.1: Acceptance Testing

Test ID	Test	Pass Criteria
AT1	Compare the micro-mouse's decision in the maze to the expected decision based on the optimization algorithm at each pixel.	The micro-mouse makes the same decision (turn left, go forward, turn right or turn around) in the maze as the expected decision.
AT2	Place a wall to the right, in front and behind the micro-mouse. Observe the micro-mouse.	The micro-mouse turns left. It remains in the black line indicating it was a 90° turn.
AT3	Place a wall to the left, in front and behind the micro-mouse. Observe the micro-mouse.	The micro-mouse turns right. It remains on the black line indicating it was a 90° turn.
AT4	Place a wall to the left, in front and to the right of the micro-mouse. Observe the micro-mouse.	The micro-mouse turns around. It remains on the black line indicating it was a 180° turn.
AT5	Create a simulation of the maze and calculate the shortest path(s) to the destination. Place micro-mouse in the physical maze and observe the path taken.	The micro-mouse takes (one of) the shortest path(s) to the destination.
AT6	Time the micro-mouse and compare to previous times taken to solve the maze.	The micro-mouse solves the maze quicker than the previous times.

4.2 Results Analysis

The results of the acceptance tests from Table 4.1 are relayed in Table 4.2. The outcome of each test is reviewed in the following subsections, with careful reference to the design elements. This analysis serves to provide an evaluation of the design previously discussed in chapter 3.

Table 4.2: Acceptance Testing Results

Test ID	Test	Result
AT1	Compare the micro-mouse's decision in the maze to the expected decision based on the optimization algorithm at each pixel.	Pass
AT2	Place a wall to the right, in front and behind the micro-mouse. Observe the micro-mouse.	Pass
AT3	Place a wall to the left, in front and behind the micro-mouse. Observe the micro-mouse.	Pass
AT4	Place a wall to the left, in front and to the right of the micro-mouse. Observe the micro-mouse.	Pass
AT5	Create a simulation of the maze and calculate the shortest path(s) to the destination. Place micro-mouse in the physical maze and observe the path taken.	Pass
AT6	Time the micro-mouse and compare to previous times taken to solve the maze.	Pass

4.2.1 Results Analysis AT1

This test involved the Decision Making design from subsection 3.1.1, the Maze Solving Algorithm from subsection 3.2.1, and the Speed design from subsection 3.2.2. At the junction of every pixel, the micro-mouse would consistently slow down which confirmed that the junction had been detected. Once the junction was detected the micro-mouse moved in the assigned direction (left, forward, right, backward) based on the Maze Solving Algorithm. The design is successful and the test passed.

4.2.2 Results Analysis AT2

This test involved the Turning design from subsection 3.1.2. Under the test conditions, the micro-mouse made a 90° left turn and continued to follow the black line. The design is successful and the test passed.

4.2.3 Results Analysis AT3

This test involved the Turning design from subsection 3.1.2. Under the test conditions, the micro-mouse made a 90° right turn and continued to follow the black line. The design is successful and the test passed.

4.2.4 Results Analysis AT4

This test involved the Turning design from subsection 3.1.2. Under the test conditions, the micro-mouse made a 180° U-turn and continued to follow the black line. The design is successful and the test passed.

4.2.5 Results Analysis AT5

This test involved the Maze Solving Algorithm design from subsection 3.2.1. The micro-mouse made the correct decisions consistently as it traversed the maze. On the first run, it starts from the corner and travels through the maze until it finds the destination pixel. On the second run, it returns to the starting corner, searching for potentially any faster routes as it goes. On the third and final run, it travels to the destination pixel along the fastest possible path. The resulting route of the micro-mouse's third route can be seen in Figure 4.1. The algorithm is successful and the test passed.

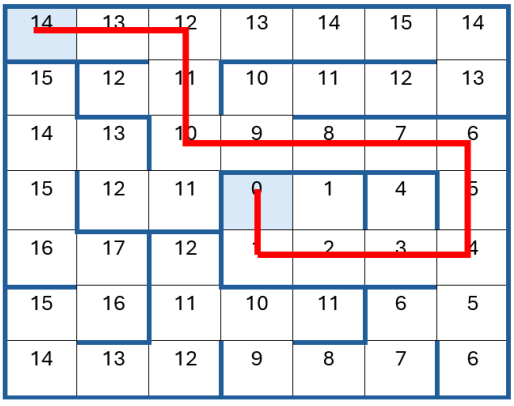

Figure 4.1: Final Run of the Micro-Mouse

4.2.6 Results Analysis AT6

This test involved the Speed design from subsection 3.2.2. The chosen speed values for the micro-mouse result in it performing consistently with relatively high speed. The micro-mouse was able to solve the maze in an adequate time of approximately 22 seconds. The design is successful and the test passed.

Chapter 5

Conclusion

All requirements for Milestones 3 and 4 were met in the developed design. The micro-mouse can accurately navigate the maze by following lines and then making decisions when it reaches line junctions. The possible decisions included, turn left, right, go straight or turn around and it could successfully determine and implement the correct decision at each junction to avoid a wall collision. Additionally, the decisions made at each junction were optimised correctly so that the micro-mouse would choose the decision that avoided any collisions and resulted in it reaching its destination point with the shortest distance using its memory of the maze. The result is that when the micro-mouse is placed in the maze, it consistently reaches its designated solution point and uses any knowledge of the walls that it acquired to determine if there are any faster routes that it could take. There are however recommendation still some recommendations that could improve the micro-mouse's current abilities.

5.1 Recommendations

5.1.1 Gyroscope

Although the micro-mouse's turns were fairly consistent, they would occasionally deviate too much, causing the micro-mouse to get lost. Implementing the onboard gyroscope could help improve the precision of the micro-mouse's turns. The gyroscope was not chosen due to it's increased complexity, however, it could still yield more accurate results than the turning and line following currently implemented. The gyroscope can work in conjunction with PID (Proportional-Integral-Derivative) control to maintain accurate line tracking and ensure that the micro-mouse turns correctly at junctions. This will lead to more consistent behaviour, especially in more complex maze sections where precise control is critical.

5.1.2 Time Optimization

The current optimization routine implemented focuses on reducing the distance travelled by the mouse. Shifting the focus from minimizing the distance to minimizing the time taken could result in better performance. Instead of strictly following the shortest path, the micro-mouse can be programmed to additionally consider paths with fewer turns, which could decrease the travel time even if the distance is slightly longer. A zigzag path, for example, may confuse the current algorithm, because it is the same length as two straight lines with a single corner to the destination, but the algorithm cannot differentiate the two and may choose the zigzag. Recording the time taken for various paths could provide be used for better optimizing the speed.

5.1.3 Turning Optimization

The current turning behaviour involves stopping at each corner before rotating, which adds unnecessary time. By optimizing the turning mechanism so that the micro-mouse takes corners gradually without stopping, the overall time taken to navigate the maze could be reduced. This smooth turning technique would allow the micro-mouse to maintain a higher speed throughout the maze, but would also mean that the current method of using junctions to determine the micro-mouse's relative position would not work.