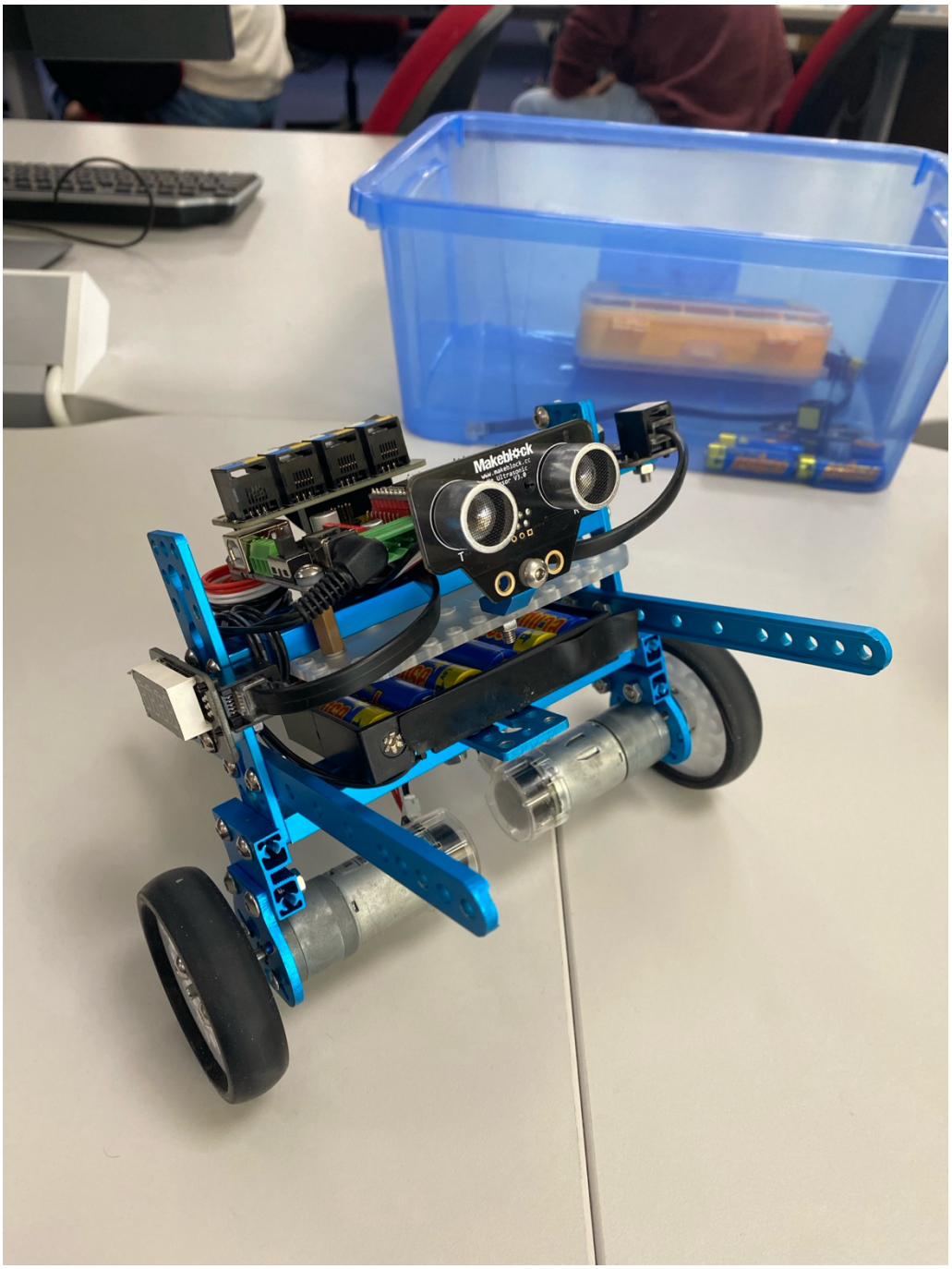
ENGMT380 Lab 1: Self-Balancing Robot



MINHO – Designed by Andrew Sun

Abstract

This report explores the design, development, and testing of MINHO, a compact self-balancing robot, with a focus on mechanical efficiency, balance, and obstacle navigation. Key design choices include reducing the mass and lowering the center of mass to improve stability, as well as implementing long arms to mimic tightrope walking, enhancing the robot's balancing ability. Using sensors like the gyroscope and ultrasonic sensor, MINHO demonstrates success in tasks involving balance, obstacle avoidance, and maze navigation. Challenges such as sensor drift and program reliability were addressed with adjustments to PID control, sensor placements, and motor constraints, leading to optimized performance. Limitations and areas for improvement, such as dynamic Z-axis adjustments and recalibration techniques, are also identified for future iterations.

Introduction

Self-balancing robots are a popular subject in robotics research due to their dynamic stability and potential for various applications, including autonomous navigation and obstacle avoidance. This report details the development and testing of MINHO, a self-balancing robot designed to complete three core tasks, each presenting unique challenges.

The first task focuses on motor control, specifically enabling the robot to maintain its balance autonomously. The second task builds on this foundation by introducing additional functionality, using sensors and actuators for tasks such as obstacle avoidance, line following, or joystick control. These tasks required teams to adopt different sensing configurations to ensure varied approaches across team members.

The third and most complex task involves navigating a maze autonomously. The robot must move through the maze based on sensor feedback, dynamically adjusting its path to avoid obstacles. This task was carried out in the D.2.03 maze, where each team member began from a different starting point. The constraints of varying sensor configurations and starting positions added complexity to the problem, requiring careful consideration of control systems and sensor integration.

This report outlines the design, mechanical and control considerations, sensor utilisation, and the methods employed to ensure MINHO's successful completion of the tasks. Key areas for future improvement are also discussed, particularly related to sensor drift and recalibration strategies during navigation.

Method and Discussion

1. Design

During the design phase of MINHO, we took various mechanical considerations into account when developing the structure of the robot. We decided to utilise a compact design, aiming to minimise mass and width. Moreover, lowering the centre of mass was an important part of the design.

The reasoning behind reducing mass was to reduce the required torque of the motors whenever MINHO would go out of balance. This would ensure that the motors were unlikely to reach stall torque, and because the motors would not need much speed and voltage, this would lower their current draw. Therefore, a lower mass contributed to the robustness of the overall mechanical design and greater versatility for the programming of MINHO. Constraining MINHO's width was crucial in making it better suited for its application in Task 3. This is because it allows MINHO to have a greater, acceptable margin of error. In the final task, MINHO showcased its ability to traverse across long sections of the maze with imperfect Z-axis angles (directions of travel).

We decided to keep the heaviest component being the battery pack at the bottom of the robot to lower its centre of mass. Furthermore, we kept the 3 components with the most significant weights in line vertically with the pivot point. Centering and lowering the center of mass of a self-balancing robot enhances its stability by reducing the torque generated by gravitational forces when the robot tilts. Mathematically, the torque τ due to gravity is given by $\tau = m \cdot g \cdot L \cdot \sin(\theta)$, where m is the mass of the robot, g is the acceleration due to gravity, L is the distance between the centre of mass and the pivot point, and θ is the tilt angle. By lowering L, the torque τ is reduced, which decreases the angular acceleration α experienced by the robot, as described by Newton's second law for rotation $\alpha = \tau / I$, where I is the moment of inertia (Ding, Y, 2012). A lower angular acceleration means the robot is less prone to rapid tipping, giving the control system more time to apply corrective actions. This results in smaller corrective forces being required, which reduces oscillations and leads to more precise balancing. Additionally, the lower torque demand reduces the energy consumption of the motors, enhancing overall efficiency. Thus, theoretically, lowering the centre of mass improves the robot's ability to maintain balance by minimising the destabilising effects of gravitational torque.

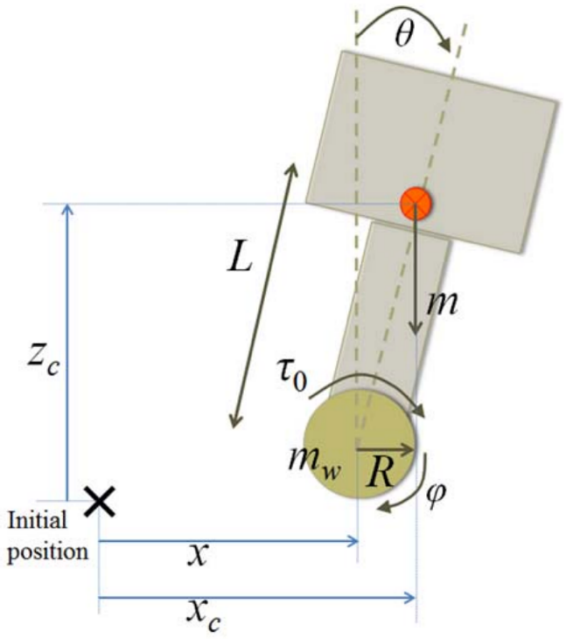


Figure 1: Parameters involved in balance

Retrieved from:

We incorporated long arms into the design to mimic a tightrope walker. Usually, a tightrope walker would have a long pole that they would hold to help them balance. This makes it easier for them to correct their centre of mass when it goes out of line with their pivot point. MINHO's long arms placed near its wheels not only increases its rotational inertia, but also places more weight near the bottom of the robot, lowering centre of mass (ScienceABC, 2023). Hence, whenever disturbances would occur and a torque is produced, due to the centre of mass moving out of line with the pivot point, this would only have a small impact on MINHO's rotation or imbalance. Long arms ensured a better overall balance of MINHO's structure and made the PID control easier to program and tune.

Overall, a small, compact design was achieved. We were able to do so while permitting an adequate amount of space to fit all the sensors and actuators. We included beams on the side to support the MegaPI board and ultrasonic sensor. Furthermore, it allowed for efficient space utilisation as we were able to fix the gyro sensor, 7-segment display, as well as any extra sensors required for tasks 1 and 2, with space set aside on the beams.

2. Sensors and Actuators

The sensors we have used throughout the tasks and their mounting locations are outlined in *Figure 2*. The gyro sensor was used to measure MINHO's angle of tilt and direction of travel. We placed the sensor in such a way that the Y-axis angle was the

angle of tilt, and the Z-axis angle was the direction of travel. The ultrasonic sensor was used the measure the distance between MINHO and any objects in front of it. The 7-segment display, although, not crucial for the physical movements of the robot, played an effective role in programming and testing. If we ever wanted to check our program to see whether it was executing as intended, we could put any sensor measurement or PWM input on the 7-segment display.

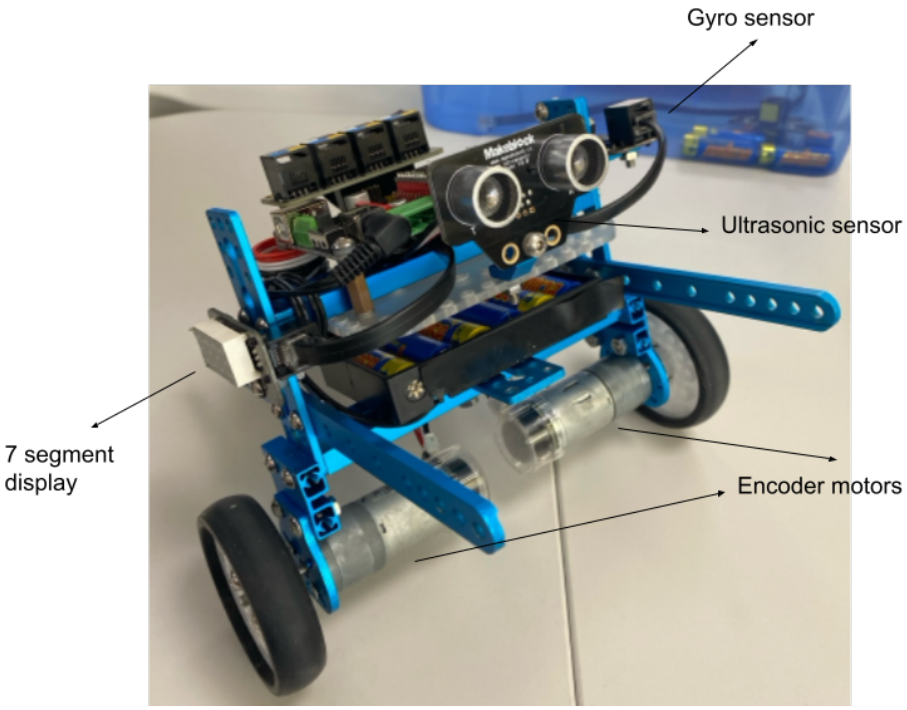


Figure 2: Sensor and actuator diagram of MINHO

The approach for completing task 1 was to collect data from the gyro sensor and calculate the error term by obtaining the difference between the measured Y-axis angle and the setpoint. We then implemented PID control, based on the error term, to control the PWM that we input into motors. To tune the PID gains, we first started by increasing the proportional gain till MINHO could oscillate about the setpoint angle for a little while. We then introduced the derivative gain to reduce oscillations allowing it to balance and counteract disturbances. The integral gain was introduced last to ensure that the robot could retain balance after an error that lasts for a long period of time. The system logic for task 1 is shown in *Figure 3* below. We found that sometimes our robot would stop functioning during the executed program. This was likely due to either memory overload, too much current draw, or short circuiting in one of the electronic components. To avoid too much current draw from the motors due to high PWM values, we constrained the PWM input to 200. We also constrained the integral gain, otherwise the integral gain would gradually increase causing a memory overload. Moreover, for sensors, we used plastic spacers to sufficiently insulate the sensors. These modifications allowed the MINHO to operate consistently and without failure.

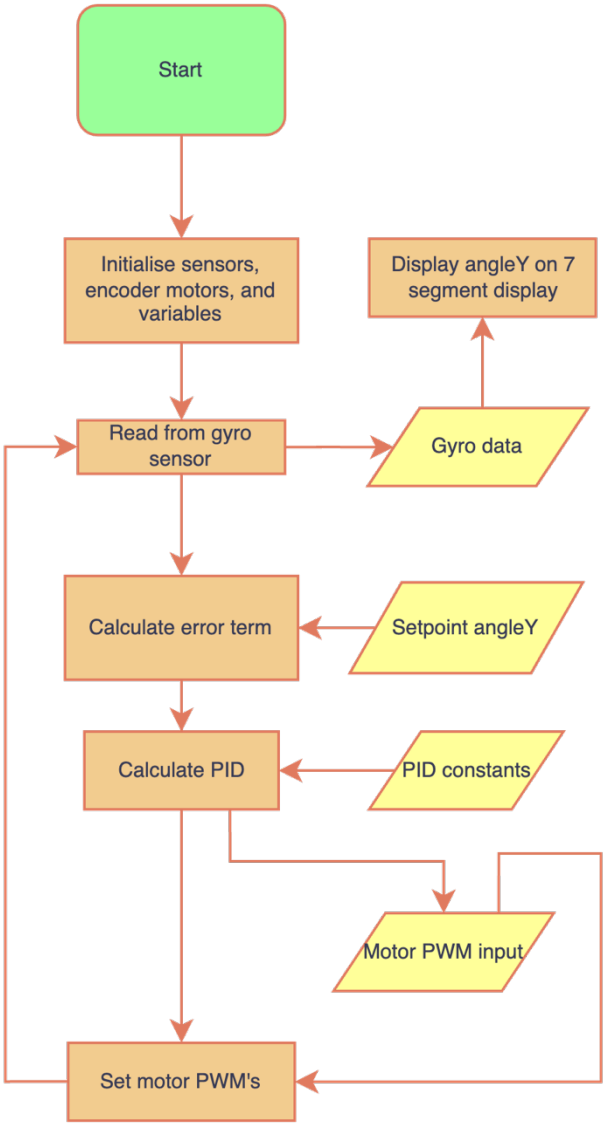


Figure 3: Flowchart of system logic for task 1

For task 2, we decided to use the Ultrasonic sensor to introduce obstacle avoidance to MINHO. To make the robot move forward we just shifted the Y-axis setpoint angle slightly forward. Before setting the PWM's for the motors, the program first collected the measured distance between the object and the robot from the ultrasonic sensor. It then checked whether the distance was less than 25 cm. If it was, the program would input the PWM's to the motors and increase the PWM of one of the motors and decrease the other by a small amount. This results in MINHO turning whenever it would detect an object in front of it. However, we encountered a slight problem. Taking data from the ultrasonic sensor every loop through the program caused MINHO to suddenly stop functioning. This was due to the high current draw from the ultrasonic sensor. To solve this issue, we introduced a delay on the ultrasonic sensor measurement. We

achieved this by setting a time delay between ultrasonic sensor readings and calculated the elapsed time since the last ultrasonic sensor reading.

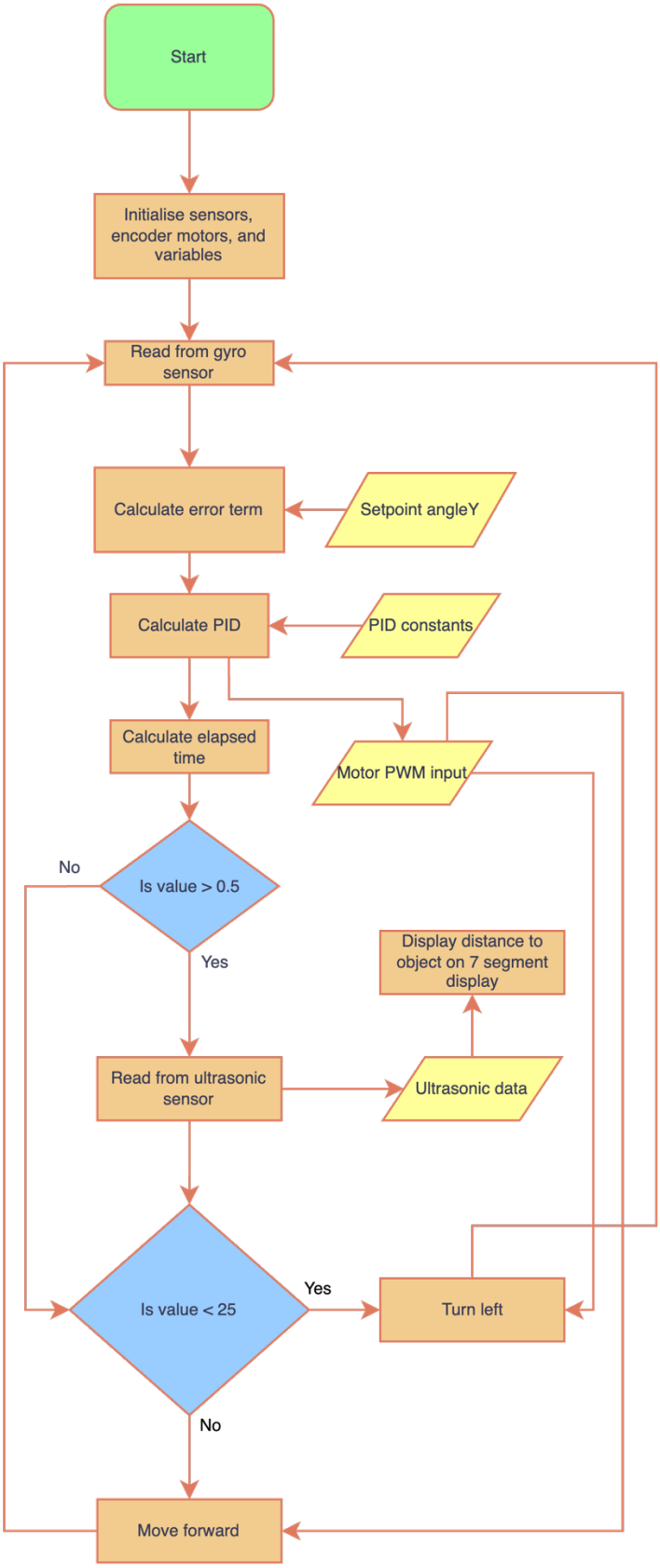


Figure 4: Flowchart of system logic for task 2

To achieve task 3 and successfully navigate through the maze, the setpoint for the robot's tilt was adjusted dynamically based on whether the robot is moving forward or executing a turn. When the ultrasonic sensor detects an obstacle within a specified distance (25 cm), the robot initiates a turn according to the predefined direction sequence (direction[]). The gyroscope provides fast feedback on the robot's Y-axis angle, allowing the robot to execute approximate 90-degree turns by controlling the motor speeds.

The gyroscope and ultrasonic sensor are the critical components that enable the robot to balance and navigate through the maze. The gyroscope ensures the robot remains upright, while the ultrasonic sensor detects walls and triggers direction changes. The encoder-equipped motors are essential for executing these movements with the necessary precision. For optimal loop speed for the program, I used a turning function that the program would enter whenever the ultrasonics sensor detected a nearby wall. If this condition was not met, the program would carry on moving forward in the main loop.

In the maze for this specific application, there were multiple factors that made the task more difficult. The carpet and slant on the floor would cause the robot to turn and drift in certain directions. To solve this problem, we also used the Z-axis angle from the gyro sensor. We used a predefined angle array so that after each direction, MINHO would follow a specific Z-axis angle to drive straight and at a path that would overcome the slants in the maze. This took a lot of testing to figure out every Z-axis angle for the robot to successfully navigate through the maze.

This approach ensured that MINHO could navigate through the maze while continuously adjusting its balance and direction, making real-time decisions based on sensor inputs. The approach is outlined in *Figure 5* below.

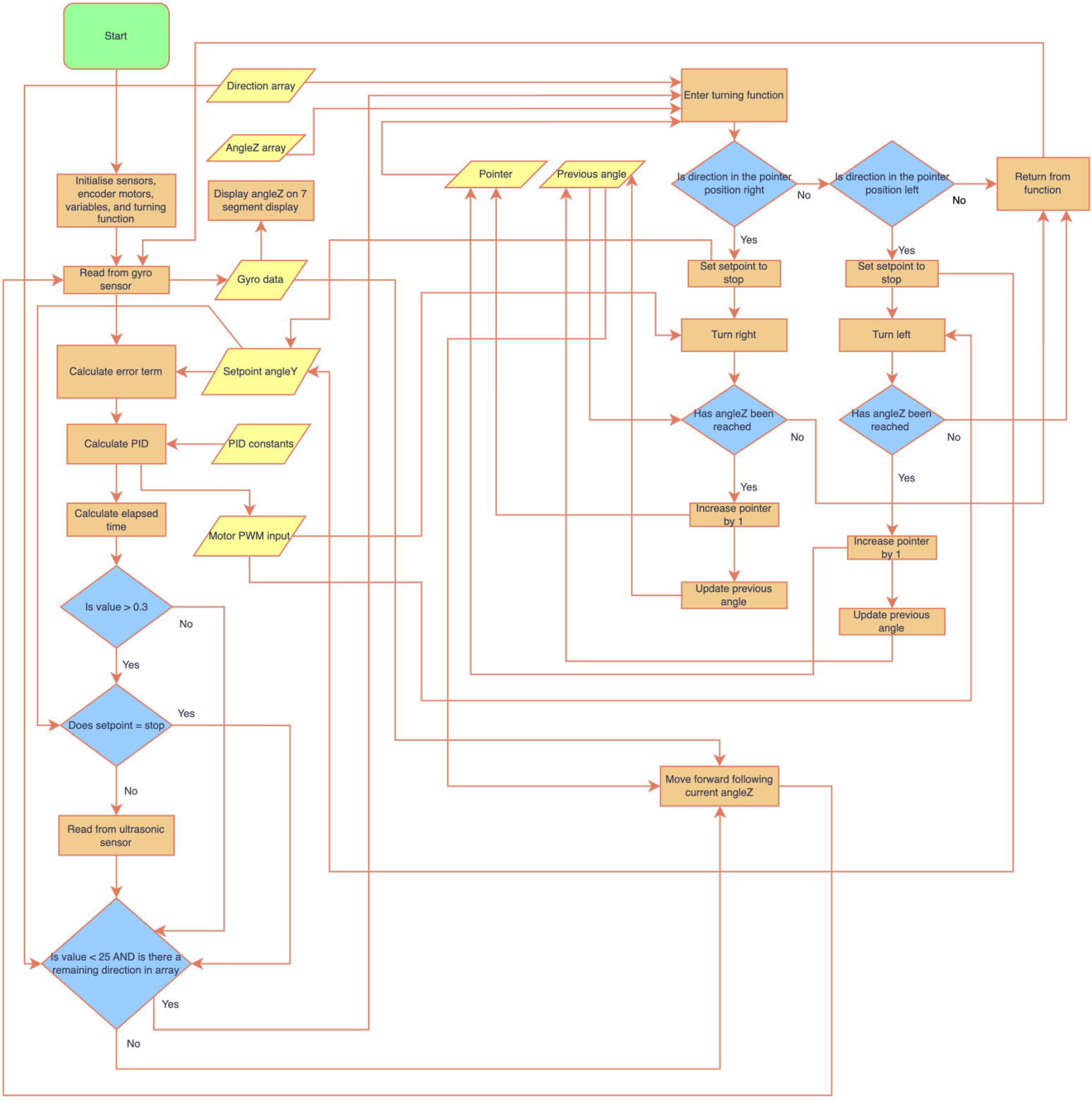


Figure 5: Flowchart of system logic for task 3

3. Limitations

Due to the predefined Z-axis angle array, this meant that the success of the run through the maze would significantly rely on the starting position of the robot. Furthermore, throughout the run, the gyro sensor would stray from its initial readings, causing the robot to traverse in non-ideal angles. The claw fixed at the rear end of MINHO made the forwards movement and the turning difficult. This is because, whenever the program would set the Y-axis angle to the stopped position, the robot would start to move backwards whilst turning.

To improve the robot's performance in navigating the maze, several adjustments could be made. Firstly, implementing a feedback loop with another sensor, collecting measurements from the side of MINHO. This is to adjust the Z-axis angle dynamically, rather than relying on a predefined array, and hence, increasing the robot's adaptability to slight variations in its starting position and ensure more accurate turns throughout the maze. Secondly, we could recalibrate the gyro sensor periodically during the run, by turning right to check its distance from the wall. This might be able to mitigate the drift from initial readings, maintaining more accurate angle measurements. Lastly, repositioning the claw to minimise its interference with forward movement and turning could enhance the robot's maneuverability, particularly when transitioning from a stopped position to turning. This could prevent the unintended backward movement that currently hampers navigation efficiency.

Conclusion

The development of MINHO provided valuable insights into the mechanics and control of self-balancing robots. By reducing the robot’s mass and lowering its centre of mass, along with incorporating long arms to enhance rotational inertia, MINHO was able to achieve a stable and compact design. The use of PID control, coupled with gyro and ultrasonic sensors, enabled the robot to balance effectively, avoid obstacles, and navigate a maze with moderate success. However, limitations such as sensor drift and predefined angle arrays affected performance, suggesting that future improvements could focus on dynamic recalibration techniques and real-time sensor feedback. Overall, this project showcases the importance of design optimization, sensor integration, and control algorithms in the creation of self-balancing robots.

Self-Balancing Robot - Core Specifications

Component	Specifications
Controller Board	
Model	Makeblock MegaPi
Microcontroller	ATmega2560
Communication	UART, I2C, SPI
Input Voltage	6V to 12V DC
Operating Voltage	5V
Current Requirement	2A to 5A (depending on load)
Power Ports	4 DC Motor Ports, 4 Encoder Motor Ports
Operating Temperature	-40 °C to 85 °C
Actuators	
Motor	Makeblock DC Motors with Encoder
Operating Voltage	6V to 12 V DC
Current Requirement (per motor)	0.5A to 2A (peak)
Maximum Torque	1.2 Nm
Encoder Resolution	360 counts per revolution
Display	Me 7-segment Display
Communication Protocol	DDI
Operating Voltage	5V DC
Operating Temperature	-40°C to 85°C
Sensors	
IMU Sensor	Me 3-Axis Accelerometer and Gyro Sensor
Communication Protocol	I2C
Operating Voltage	5V DC
Operating Temperature	0°C to 70°C
Digital Sensor	Me Ultrasonic Sensor
Communication Protocol	SDI
Operating Voltage	5V DC
Operating Temperature Range	-25°C to 80°C
Communication	
Bluetooth Communication	Bluetooth 4.0
Wi-Fi Communication	Optional (via add-on module)
Wired Communication	USB 2.0, Serial
Motor Driver	Me Encoder Motor Driver
Communication Protocol	I2C
Operating Voltage	6V to 12V DC
Operating Temperature	-40°C to 85°C

References

ScienceABC. (2023, October 19). *Why tightrope walkers carry a pole during their performance?*. https://www.scienceabc.com/eyeopeners/why-tightrope-walkers-carry-a-polebar-during-their-performance.html

Ding, Y., Gafford, J., & Kunio, M. (2012, December 18). *Modeling, Simulation and Fabrication of a Balancing Robot*. *https://scholar.harvard.edu/files/jgafford/files/finalpaper_final_version.pdf*