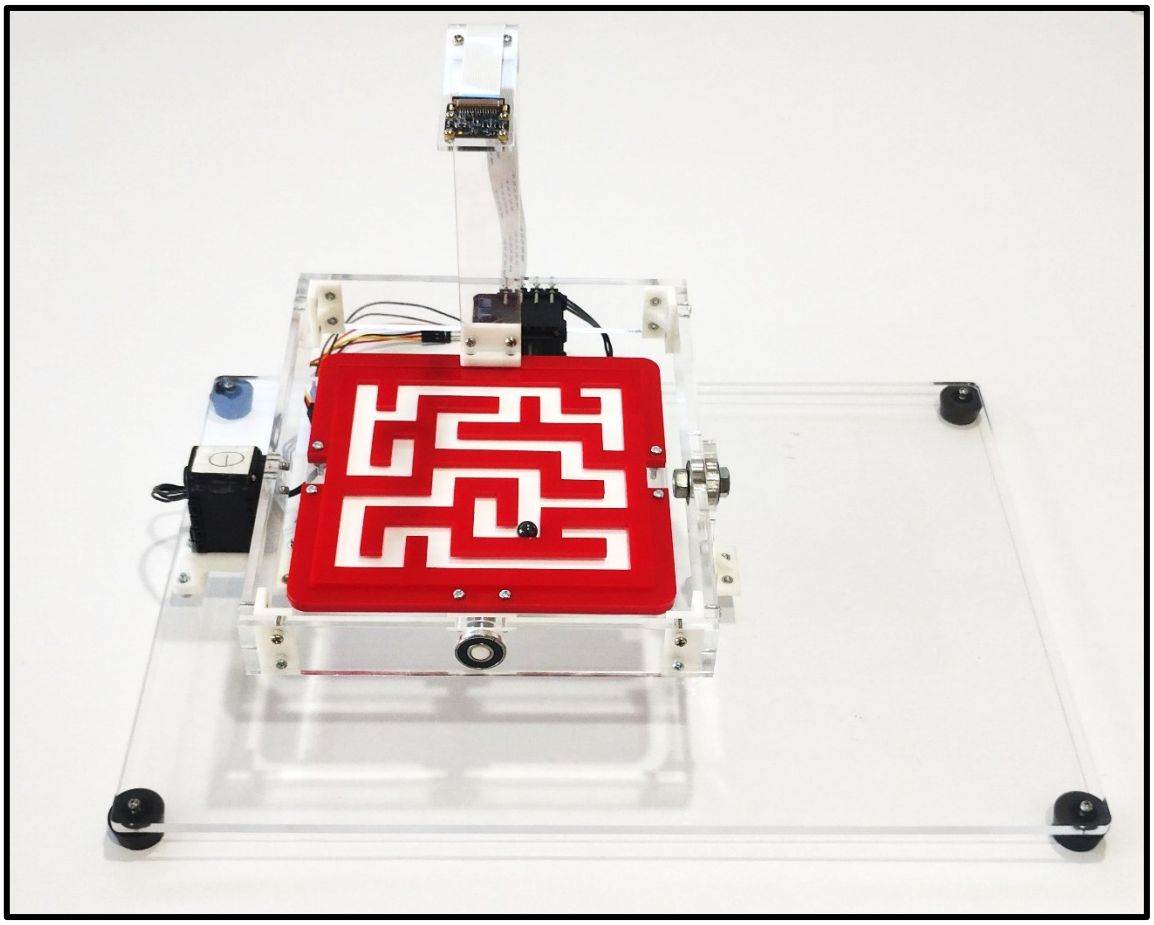
MONASH University
Engineering



Ball-In-Maze Solving Robot

ENG4702: Final Year Project - Final Report

Author(s): Avvienash Jaganathan (32281013)

Supervisor(s): Michael Zenere

Date of Submission: 24th May 2024

Project type: Consultancy

Table of Contents

1 Executive Summary

The Ball-in-Maze is a hand-eye coordination puzzle with the objective of manipulating a maze, usually by tilting it, to guide a ball towards a goal. This project aims to develop a vision-based maze-solving robot capable of autonomously controlling a ball within a maze. The system utilizes image processing, control techniques, and maze-solving algorithms for educational, recreational, and research purposes.

The final product uses a 2-degree-of-freedom tilting platform, similar to a two-axis gimbal, to control the maze's orientation. A top-down camera accurately detects the ball's position and the maze layout. The system integrates a Raspberry Pi for high-level decision-making and PSOC microcontrollers for servo control. Path planning algorithms, such as RRT*, generate optimal paths, while PID control is used to control the ball's movement.

A web-based user interface allows for intuitive interaction and real-time monitoring. Users can initiate maze solving, monitor progress, and cancel operations as needed. During testing, the system demonstrated high reliability, achieving a 100% success rate in navigating a ball through various maze designs. The system also features interchangeable maze designs, promoting user creativity and engagement without hardware modifications.

Despite the project's success, several areas for improvement were identified. The PID controller showed limitations in complex scenarios, leading to higher solving times and occasional path deviations. Implementing more advanced control algorithms, such as Model Predictive Control (MPC), could enhance the system's responsiveness and accuracy.

Overall, this project successfully designed and prototyped a flexible and user-friendly autonomous ball-in-maze-solving robot with potential for future improvements.

2 Acknowledgements

The project is being held on the lands of the people of the Kulin Nations and I wish to acknowledge them as Traditional Owners. I would also like to pay my respects to their Elders, past and present.

Importantly, this project would not have been possible if not for my project supervisor, Michael Zenere, for his guidance, help with procuring parts, and debugging. Special thanks to Rob Jackal for design considerations and assistance with 3D printing and laser cutting. Lastly, thank you to the Department of Electrical and Computer Systems Engineering (ECSE) at Monash University for providing support and resources throughout this project.

3 Introduction

The Ball-in-Maze is a puzzle of hand-eye coordination skill consisting of a maze and one or more balls. The objective of the game is to manipulate the maze, usually by tilting it, to guide the ball towards the goal without letting it fall through any of the holes as shown in **Figure 1 (Left)** [1]. It has been hypothesised to be the oldest dexterity puzzle in the world as a similar artefact was found in the Indus Valley Civilisation around 2500 BC as shown in **Figure 1 (Right)** [2] [3].

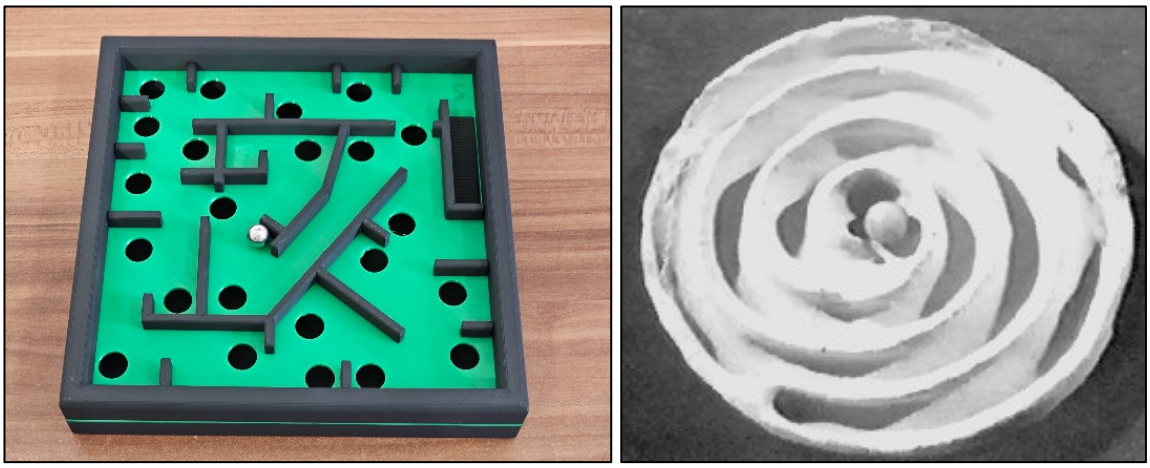


Figure 1: Modern Ball-In-Maze Puzzle (Left) and Ancient Dexterity Puzzle (Right).

A Ball-In-Maze solving robot is educationally significant as it provides a firsthand learning experience for students, engineers, and robotics enthusiasts. It is a practical tool for learning various aspects of robotics systems, computer vision techniques, and control algorithms. By working with such a robot, learners can gain valuable insights into the integration of hardware and software systems, the implementation of real-time image processing, and the development of sophisticated control strategies to navigate the Maze [4].

Beyond education, it is used for research purposes, investigating various algorithms and control systems. For instance, a recent conference paper was published on Dual arm manipulation and whole-body control with the humanoid robot Romeo that solves a ball-in-maze game in augmented reality using only vision and two hands [5]. This robot can serve as a base to test new algorithms, image processing techniques, and compare different approaches to in maze solving problems.

Automated game solvers like the Ball-In-Maze solving robot have recently become popular for recreational and stress-relief purposes. These robots offer an engaging and interactive experience that not only entertains but also helps reduce stress and improve cognitive abilities [6] [7].

In summary, this project seeks to develop a vision-based maze-solving robot capable of autonomously controlling a ball within a maze, using image processing, control techniques, and maze solving algorithms for educational, recreational and research purposes.

4 Aims and Objectives

4.1 Problem Opportunity Statement

4.1.1 Current State

Existing Ball-In-Maze Solving Robots struggle in achieving fast and precise control over the ball's movement within the maze, resulting in inefficient maze-solving processes. These robots are limited to a simple maze layout such as grid-based mazes and lack adaptability to manage multiple or changing maze configurations. Additionally, the existing robots lack a user interface or GUI to visualize the maze-solving process, making debugging and fine-tuning difficult and unintuitive.

4.1.2 Impact

Due to the limited precision in controlling the ball and lack of adaptability in the maze layouts, their current practical applications are limited to tabletop toys. The untapped potential for advanced maze-solving capabilities could benefit various fields such as education and research. Furthermore, the lack of a user interface hampers visualization, debugging, and fine-tuning processes.

4.1.3 Desired State

The desired state is a Ball-In-Maze Solving Robot that has a higher precision in controlling the ball's position for a more efficient and accurate maze-solving. This robot should be capable of solving mazes with intricate designs and varying levels of complexity, as well as adapting to dynamic environments. The implementation of a user-friendly interface for visualizing the robot's progress and state will enhance usability and improve the user’s experience. Additionally, the robot should implement robust safety mechanisms to ensure safe operation during maze-solving tasks.

4.2 Aim

The aim of this project is to develop a vision-based maze-solving robot capable of autonomously controlling a ball within a maze, using image processing, control techniques, and maze solving algorithms to achieve accurate and efficient navigation. This robot will showcase the capabilities of robotics systems for educational, research and recreational purposes. Furthermore, this project aims to make this robot as affordable, sustainable, and safe as possible.

4.3 Objectives

Based on the aim, objectives for the project are highlighted in **Table 1**. The objectives are ordered chronologically by accounting for its dependencies and prerequisites. The dates are in Monash Semester Format.

Table 1: Project Objectives.

No.	Objective	Success Criteria	Estimated Completion
1	Develop a vision-based control system using advanced image processing to detect the ball's position and maze configuration in real-time.	System accurately detects the ball and maze with at least 95% accuracy in real-time.	Week 12 Semester 2 2023
2	Implement sophisticated control algorithms to achieve precise movement and positioning of the ball within the maze.	Ball reaches the designated goal in the maze within a tolerance of ±5 mm.	Week 3 Semester 1 2024
3	Ensure the robot can handle mazes with intricate designs and varying levels of complexity and adapt to dynamic environments.	Successfully solve mazes with at least 3 varying complexity levels and configurations	Week 6 Semester 1 2024
4	Create a graphical user interface (GUI) for real-time visualization of the robot's progress and state.	User interface updates in real-time with less than 0.5-second latency.	Week 8 Semester 1 2024
5	Implement safety features to prevent accidents or damage during the robot's operation.	Perform Safety testing and ensure all test cases are passed.	Week 10 Semester 1 2024
6	Focus on making the robot cost-effective without compromising functionality and performance.	Maintain cost under AUD 150 while meeting all functional requirements.	Week 12 Semester 1 2024

5 State of the Field

5.1 System Model and Environment

The Ball in Maze Model (BiM) is a system designed to control the position and trajectory of a ball on a maze. The goal is to navigate the ball through a maze constructed on the plate using a closed-loop control system and image processing techniques. This model can be simplified to a similar model such as the Ball on Plate Model which is a two-dimensional system designed to control the position and trajectory of a ball on a plate. The same principles can be applied in a BiM as it is the same model with a maze mounted upon the plate [8] [9].

Figure 2 shows the System and Environment of the Ball on Plate model.

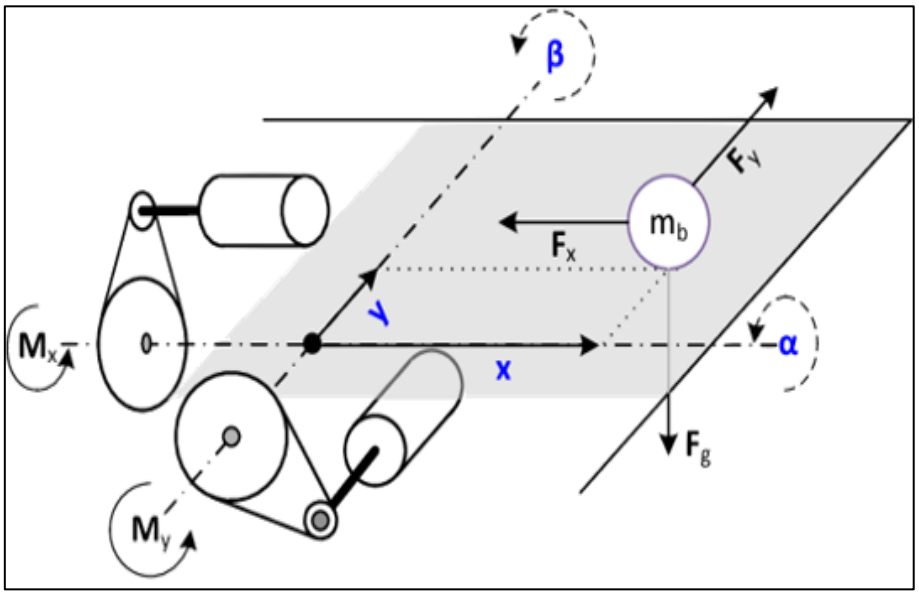


Figure 2: Ball on Plate model diagram.

Table 2: Analysis on Ball on Plate Model

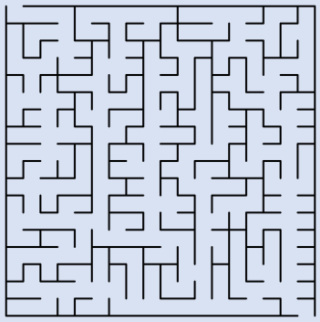
Aspect	Ball on Plate Model	Implications for BiM
Purpose	Control ball position and trajectory on a plate	Concept can be extended to controlling ball in a maze
Principle	Closed-loop control and image processing	Similar principles can guide ball movement in a maze
Environment	Management of ball movement on a surface	Maze mounted on a plate creates the same environment
Control Mechanism	Two-dimensional ball manipulation	Two-dimensional ball manipulation

Table 2 highlights the relevance of the Ball on Plate Model for the BiM. Therefore, we will treat the BiM as an extension of the **Ball on Plate Model**.

5.2 Wall Structure

Two main structure types were used in the development of the Ball in Maze Walls with thickness and walls without thickness. A comparative table highlighting the distinctions between the two designs is shown in **Table 3**.

Table 3: Comparison of Binary Grid and Wall based Structure.

Design Type	Wall with Thickness	Wall without Thickness
Description	Binary cells, where each cell can be a wall or an empty space	Edges of cells are used as walls.
Example Image		
Computational Vision Complexity	Low	High

To reduce Image processing and pathfinding complexity, we opted for the **binary grid design**. Since the Maze is Modular and can be easily interchanged, we may implement a wall-based structure in future designs.

5.3 Maze Physical Construction

The physical construction of the maze for a maze-solving robot can be built using materials like wood, plastic, cardboard, or laser-cut acrylic. The walls should be sturdy to prevent the ball from knocking them over. Some studies use a commercially available Ball in Maze Puzzle modified for robust attachment points for the servo link rods and gimbals [10].

Alternatively, some studies use artificial walls such as coloured tape, creating a 2D projection controlled solely by tilting the plate. This method is low-cost and quick to design. In this approach, the ball's movement was solely controlled by the controller via the tilting of the plate, eliminating the support from physical walls [8]. Another common approach involves using laser-cutting or 3D printing. These methods offer precision and customisability. Via this approach, we are able to design the maze according to our specification and dimensions. **Figure 3** shows a commercially available Ball in Maze Puzzle from Amazon modified with a 3D-printed plate, a maze with artificial wall constructed with blue electrical tape, an acrylic laser-cut maze and a 3D printed Maze respectively.

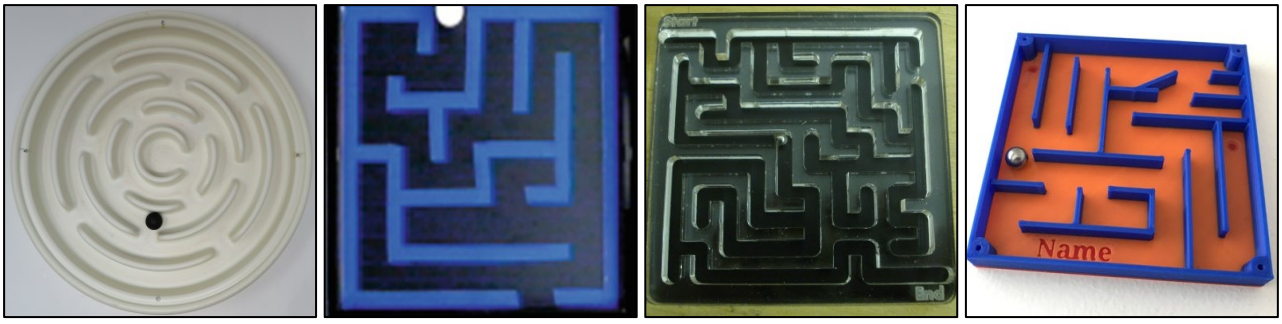


Figure 3: From Left to right: Amazon Purchased, Coloured Tape, Laser-cut, and 3D printed Maze.

Table 4: Analysis on Maze Construction Approaches

	Amazon Purchased	Coloured Tape	3D Printed	Laser-Cut
Material	Plastic, Amazon puzzle	Coloured tape	PLA	Acrylic
Construction Complexity	None	Simple (tape application)	Moderate (design and printing)	Moderate (design and cutting)
Controller Complexity	Moderate	High (artificial wall, hence controller must be very accurate)	Moderate	Moderate
Construction time	High (Wait for delivery)	Low (tape application time)	Moderate (printing time)	Low (Cutting using a laser)
Customization of Maze layout	Limited to Amazon puzzle	High	High	High
Cost	Moderate to high	Very low	Low to moderate	Low to moderate
Sustainability	Low	High (Minimal materials)	Moderate (Waste PLA)	High (recyclable)

Table 4 provides a comparison of different maze construction approaches based on various aspects such as the materials used, construction complexity, controller complexity, customization options, and cost. Therefore, for the scope of this project, physical walls will be used to reduce the complexity of the controller design. Hence, we will be using the **Laser-cut Maze** approach as it gives us more customisability at a reasonable cost.

5.4 Maze Layout Design

The layout of the maze is critical, as it affects the robot's navigation and the difficulty of the maze-solving task. The maze can be designed in various configurations, such as a simple grid-based layout, a random path, or a radial structure. Design considerations include dead-ends, loops, and the placement of the start and end points.

The complexity and difficulty of the maze can be adjusted based on the project's goals and the robot's capabilities. For educational purposes, simpler mazes may be preferred to introduce basic concepts, while more complex mazes can challenge advanced algorithms and control strategies.

To obtain consistent results, the maze was designed to have only one exit and one entrance [8]. Another consideration when designing this maze was to ensure that there were no loops in the maze. Although loops are common within the maze, having a loop would create multiple branches, confusing the algorithm.

The maze design includes the addition of holes. These holes are usually placed at specific locations within the maze, and the robot's objective is to avoid falling into them while reaching the destination. For a simpler approach, holes may be removed from the design.

Calibration patterns are also integrated into the maze design. These patterns, such as checkerboards or printed markers of known dimensions, help calibrate the robot's camera system for accurate perception of the maze environment.

Lastly, depending on the project's scope and objectives, multiple maze designs may be created. Each maze can offer various levels of difficulty and present unique challenges for the robot to solve.

Table 5 compares aspects of maze design configurations.

Table 5: Comparison of aspects of maze design configurations

Configuration	Grid-Based Layout	Random Path	Radial Structure
Design Complexity	Low	Random and unpredictable	Moderate
Path Variation	Limited	Diverse	Limited
Controller Complexity	Moderate	Moderate to high	Moderate to high
Image Processing Complexity	Low	Moderate to high	Moderate to high

Therefore, in the scope of this project, **a grid-based layout** will be used as a base when designing the maze to ensure a structured and organised layout that eases Image processing and reduces controller complexity.

5.5 Image Processing and Computer Vision

5.5.1 Camera Setup

The camera used in the maze-solving robot is typically mounted on the robot chassis or an adjustable mount to capture images of the maze from a top-down perspective. The camera's position should be such that it captures the entire maze area, including the starting and ending points. The image captured should be able to see all the walls, holes, and the ball regardless of the roll and pitch of the plate. **Figure 4** shows the camera setup for a top-down view of the maze.

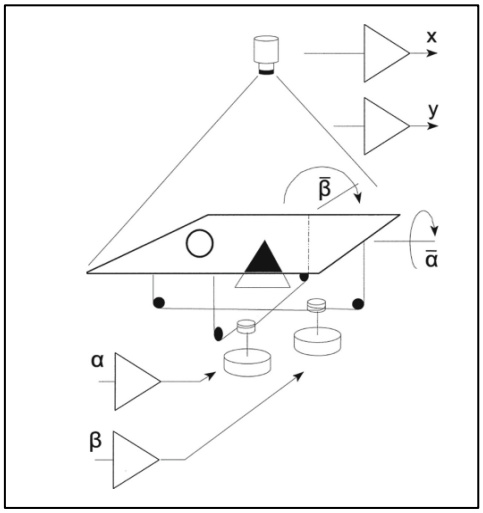


Figure 4: Top-Down view camera setup.

The captured image is in RGB colour format, representing the maze with its walls, path, and any other elements that may be present.

5.5.2 Camera Calibration

The camera's perspective introduces distortions in the images, which can affect the robot's ability to accurately perceive the maze layout and make informed decisions.

Camera calibration is a critical step in computer vision and image processing, especially when dealing with real-world images and measurements. It enables the establishment of a relationship between 2D image pixel coordinates and 3D real-world coordinates, allowing accurate mapping of image points to the physical world. The camera calibration process typically involves capturing a set of calibration images with known 3D reference points and then using these images to estimate the camera's intrinsic and extrinsic parameters.

The intrinsic parameters of the camera are inherent to the camera and remain constant for a specific camera model. They include parameters such as the focal length (fx, fy), principal point (cx, cy), and lens distortion coefficients. These parameters determine how the 3D world coordinates are projected onto the 2D image plane. The extrinsic parameters represent the camera's position and orientation in the 3D world concerning a reference coordinate system. They include the rotation matrix (R) and translation vector (T) that describe the camera's pose.

In this project, the feature detection algorithm should be capable of identifying the calibration target's specific pattern (e.g., maze patterns or features of known dimensions) in the captured images. The detected feature points will establish correspondences between 2D image pixel coordinates and 3D real-world coordinates [11] .

Hence, the robot's camera can accurately map image points to real-world coordinates. This precision is crucial for the robot to navigate the maze, detect obstacles, and make informed decisions based on the captured images.

5.5.3 Colour Segmentation

The preprocessing of images for Ball in Maze solving robots is a crucial step in the overall maze-solving process. It involves various techniques to enhance and prepare the captured maze image for further analysis and navigation.

In a study, the preprocessing technique involves segmenting the maze image based on colour spaces. In particular, the RGB colour model is used to identify blue walls in the maze [8]. The blue component of the RGB model is subtracted from the maximum of the red and green components to compute the "blueness" of the image.

$$b = B - max(R, G)$$

This blueness image is used to create a binary mask to separate the maze walls from other elements in the image. **Figure 5** shows the results from a blueness image to the binary mask.

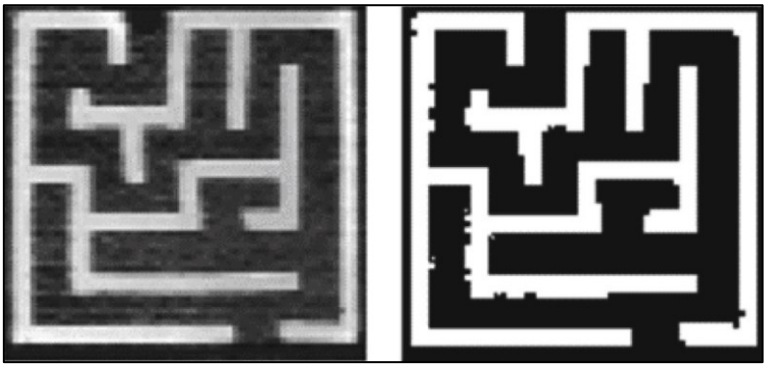


Figure 5: Blueness image (left) and its binary mask (right)

5.5.4 Watershed transformation

This project also implements the watershed transformation which is a tool used to segment grayscale images [8]. In this project, the watershed transform is used to segment the maze further and obtain distinct regions for various parts of the maze. It considers a grayscale image as a topographical relief with the intensity of the pixel corresponding to the elevation of the surface.

5.5.5 Image Enhancement

To ensure accurate image analysis, noise reduction techniques are often employed during preprocessing. Unwanted noise in the image can hinder the robot's navigation and image processing algorithms.

Techniques that can be applied include Gaussian filtering, or median filtering to remove noise and retain essential features of the maze image [10] .

5.5.6 Detection of Points on the Maze

The detection of the starting and end points of the maze is a crucial step in maze-solving robotics, as it helps the robot identify the entrance and exit of the maze. In the context of image processing, this step involves analysing the image to identify the specific markers that indicate the starting and end locations.

In a study, the maze-solving robot detects the starting location and end location of the maze via coloured markers. The starting location, which represents the maze entrance, is identified by a red-coloured line, while the end location, representing the maze exit gate, is represented by a yellow-coloured line. These coloured lines are crucial markers for the robot's navigation through the maze [11].

Table 6 highlights the purpose of the various aspects of image processing.

Table 6: Purpose of image processing and computer vision aspects

Aspect	Purpose
Camera Setup	Obtain top-down perspective of the maze for comprehensive image capture.
Camera Calibration	Correct distortions caused by camera perspective, ensuring accurate image-to-world mapping.
Colour Segmentation	Separate maze elements from the image.
Watershed Transformation	Segment grayscale maze images into distinct regions and identify distinct parts of the maze layout.
Image Enhancement	Reduce noise to enhance image quality, preserving essential maze features.
Point Detection	Identify entrance and exit points, guiding the robot's navigation through the maze.

In conclusion, these image processing techniques allow the maze-solving robot to interpret the maze environment accurately.

5.6 Ball localization and tracking

In colour-based ball detection, the robot looks for specific colour cues to identify the ball in the maze image. The ball is usually of a distinct colour from the maze walls and background. The robot applies colour thresholding to segment the image and isolate the pixels that correspond to the ball's colour [11].

After colour-based segmentation, the robot applies blob detection or contour analysis to identify connected regions of pixels that form the ball's shape [10]. Blob detection groups adjacent pixels with similar colour characteristics, while contour analysis identifies the boundaries of the ball in the image. These techniques help in accurately locating the position of the ball in the image.

Using the camera calibration parameters, the 2D pixel coordinates (x, y) of the ball are converted to real-world coordinates (X, Y, Z) in the maze platform's coordinate system. This transformation involves considering the camera's position and orientation relative to the maze platform.

Since this study uses a Circular maze environment, the real-world coordinates (X, Y, Z) are converted to polar coordinates, with 'r' representing the distance between the ball and the centre and 'θ' representing the rotation around the z-axis. This information is essential for the robot to understand its current position within the maze [10].

Table 7 compares the two coordinate systems.

Table 7: Comparison of Coordinate Systems

Coordinate System	Description	Pros	Cons
Cartesian Coordinates	Uses x and y axes to represent positions.	Well-suited for rectangular environments.	Not suitable for polar-oriented tasks.
Polar Coordinates	Utilizes radial distance (r) and angle (θ) to represent positions.	Well-suited for circular environments. Suitable for tasks involving angles.	Not suitable for grid-based tasks.

Therefore, since this project will be using a grid-based maze, **Cartesian Coordinates** will be used for mapping purposes.

To track the ball's motion, the robot's control system estimates the angular velocity (θ) of the ball. This paper used the Kalman filter, which is a recursive algorithm that estimates the state of a dynamic system based on noisy measurements over time. The angular velocity helps predict the ball's future positions based on its current motion [10].

Since the camera continuously captures images, the ball's position and velocity are continuously updated. This provides real-time information about the ball's movement and enables the robot's control system to adjust its actions accordingly.

Another Study uses a resistive touch panel to detect the actual position of the ball on the top panel. It works in conjunction with the ADCs to convert the analogue signals from the touch panel into digital values that can be processed by the microcontroller [12].

Figure 6 shows a Ball on Plate project that utilizes a resistive touch panel and a top-down camera to read the balls position respectively.

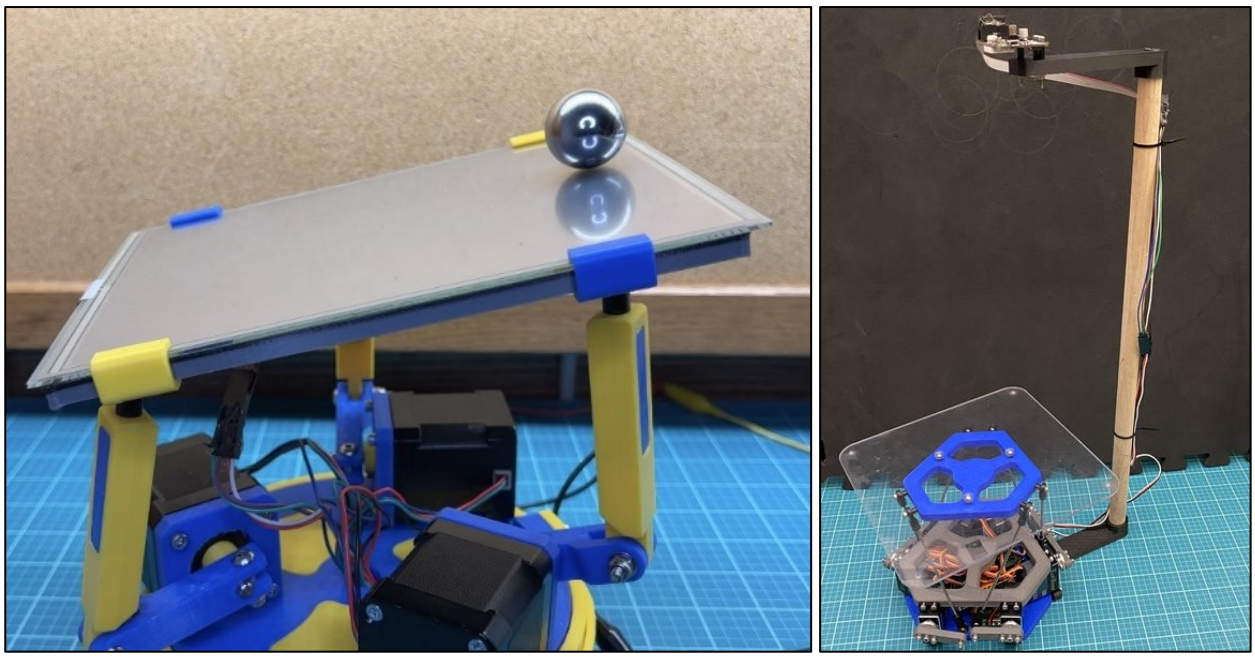


Figure 6: Resistive Touch Panel (Left) and a Top-Down Camera (Right) to read the ball's position.

Table 8 compares the two different ball tacking techniques.

Table 8: Comparison of Ball Tracking method

Technique	Method	Pros	Cons
Image-based Ball Detection	Utilizes image processing and computer vision techniques to identify the ball's position from captured images.	- Works well in visually distinct environments. - Lower cost	- Affected by lighting conditions. - Limited to visual information. - May require calibration.
Touch Panel Ball Detection	Relies on resistive touch panel to detect ball's position by physical contact.	- High accuracy in position detection.	- Requires physical contact with the ball. - May introduce friction or disturbance. - Higher Cost

Therefore, in the scope of this project, **Image based Ball detection** will be employed as image-based detection will have to be implemented anyway for the maze layout detection as well as to save cost.

5.7 Type of Robotic manipulator

Magnetic control methods are used to manipulate a spherical magnet (ball) using a computer-controlled magnetic dipole source called the Omnimagnet [12]. The Omnimagnet generates a fully controllable dipole magnetic field, which is capable of varying both its direction and magnitude. To control the ball's position and movement, they utilize the nonuniform magnetic field generated by the Omnimagnet, which exerts both forces and torques on the magnetic object (the spherical magnet). By adjusting the dipole-moment direction and magnitude of the Omnimagnet, they can control the magnetic forces and torques acting on the ball.

A parallel manipulator, also known as a parallel kinematic manipulator, is a type of mechanical system used in robotics and automation. It consists of a closed-loop kinematic chain mechanism where the end effector is connected to the base through multiple independent kinematic chains [13].

A serial manipulator is a robotic system built upon the concept of an open kinematic chain. It consists of a sequence of interconnected bodies, typically including one fixed base and one mobile end effector, with links in between. These links are arranged, often through joints, in a way that permits controlled relative motion between them. The precision of serial manipulators is derived from their ability to articulate each joint in the chain, facilitating accurate positioning and manipulation of the end effector [13][14].

Due to the excessive cost and specific environment conditions required, as well as the scope of the project requiring the maze to be manipulated instead of the ball, the Magnetic Dipole Source will not be considered as an option. Hence a robotic manipulator will be used.

The **Table 9** shows the comparison between parallel and serial manipulators.

Table 9: Comparison of Parallel and Serial Manipulators

Aspect	Parallel Manipulator	Serial Manipulator
Type of Manipulator	Closed-loop kinematic chain	Open-loop linear sequence
Advantages	High load capacity, low inertia, structural stiffness, precision	Larger workspace, dexterity, easier direct kinematics
Kinematics	Complex direct kinematics due to closed loops, simplified inverse kinematics	Straightforward direct kinematics, potentially complex inverse kinematics

Accuracy	Inherently more accurate due to averaging of errors, high stiffness, and low inertia	Prone to cumulative error accumulation, potential for vibration and bending
Preferred Application	Precise positioning, heavy load	Gross motion, larger workspaces

Therefore, in the scope of this project, we require precise positioning but do not need extensive range of motions. Hence, **Parallel Manipulators** will be used in this project.

Most robotics systems designed to solve the BiM, use a two-DOF tilting platform to control the ball. For example, this study uses an electromagnetic tip-tilt system. The system comprises the maze, purchased from Amazon, an attachment plate, two gimbals, two RC servos enabling pure rotational motion around its origin, a 2D laser tilt sensor, a base plate, an overhead camera, and an Arduino-based control processor. The laser is utilized to measure the inclination of the maze platform, and the servos are operated via the Robot Operating System (ROS). **Figure 7** shows the coordinate reference frames for the maze environment on the tip-tilt platform. Angles β and γ represent the orientation of the tip-tilt stage [10]. The main challenge in modelling the system is dealing with dry friction and contacts, which are difficult to represent accurately using conventional physical models. To address this, a paper proposes using a semiparametric model based on Gaussian Process Regression to estimate the motion dynamics of the ball.

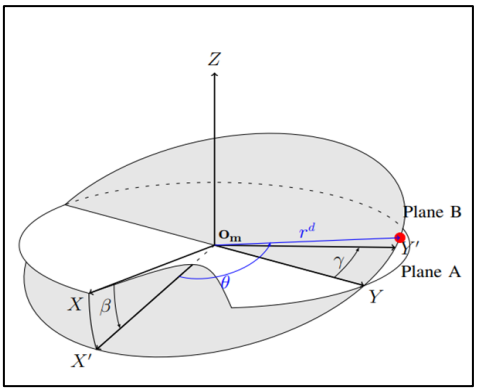


Figure 7: Two-Servo Tilting Platform Model.

Another study designed a control system for a ball on plate model without a maze. The design also involves a two-DOF system with two servo motors. The system involves two manipulated variables and two controlled variables. The two servos are controlled using Pulse Width Modulation (PWM) signals. This controls the rotation of the plate along the x and y axis [15].

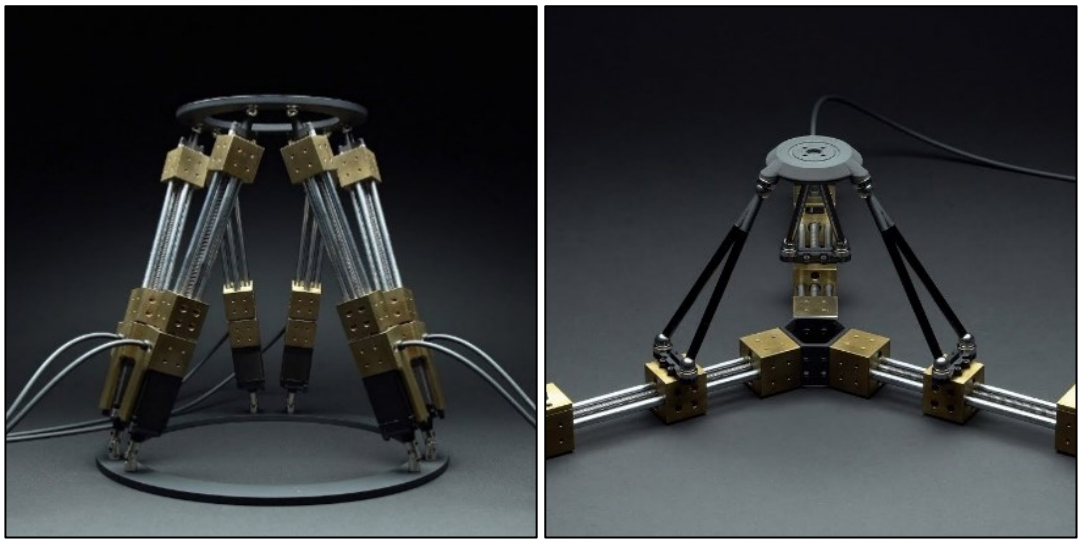


Figure 8: Stewart Platform (Left) and Delta Platform (Right).

The Stewart Platform, originally designed by Stewart, originated as a mechanism for simulating flight conditions with a 6-degree-of-freedom structure shown in **Figure 8 (Left)**. This mechanism featured a triangular platform supported by adjustable-length legs connected to the ground through ball joints. Gough later suggested using six parallel linear actuators, like a tire test machine, transforming it into a fully parallel-actuated manipulator. This manipulator has various advantages such as enhanced stiffness and precise positioning capabilities [16].

A Delta robot is a type of parallel robot constructed using parallelogram mechanisms. Its distinctive feature is a moving platform that possesses three translational Degrees of Freedom (DoFs) and one rotational DoF concerning the fixed base as shown in **Figure 8 (Right)**. The concept of the Delta robot was first introduced in a WIPO patent in 1986 by Clavel and was later elaborated upon by various researchers. These studies included deriving equations for forward and inverse kinematics, analysing the dynamics and mass matrix of the Delta robot, improving its accuracy, and even creating three-dimensional CAD simulation tools. Over time, the Delta robot concept has evolved into various parallel kinematics machines and has found applications in multiple fields. [17]

Table 10 highlights the comparisons between the manipulator designs.

Table 10: Comparison of Different Controller Designs for Ball in Maze Robots

Parallel Manipulator	Kinematics Complexity	Degrees of Freedom	Number of Actuators	Cost
Two Servo Tilting Platform	Moderate	2	2	Low
Stewart Platform	Complex	6	6	High
Delta Robot	Complex	4	3	High

Given the limited budget, the number of actuators should be minimised. Besides that, to solve the maze, only 2 DOF is required. Therefore, in the scope of this project, **two servo tilting platform** will be used to ensure consistency as well as lower the cost. There are two distinct types of two servo titling platforms, based mounted servos and frame embedded servos (two-axis Gimbal) as shown in **Figure 9** [18].

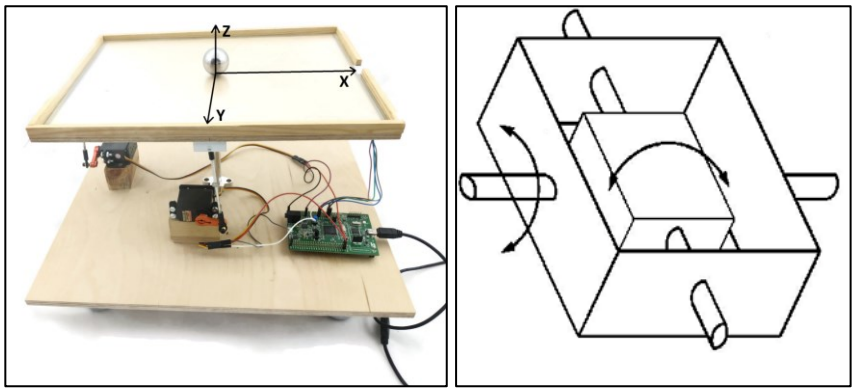


Figure 9: Base-Mounted Servos System (Left) and Frame-Embedded Servos System

Table 11 perform comparisons on both approaches.

Table 11: Comparison of tilting platform configurations

Option	Base-Mounted Servos	Frame-Embedded Servos
Servo Placement	Both servos mounted on the base	Servo 1 on inner frame, Servo 2 on outer frame
Joint Mechanism	Universal joint/ball and socket joint for platform with links on the side connected to the servos	Inner frame connected to Servo 1; Outer frame connected to Servo 2 which is connected to the base
Max Tilt Angle	Low	High
Cost	Moderate	Moderate
Control Precision	Moderate	High
Construction Complexity	Low	Moderate

Hence, after careful consideration, we have chosen the **Frame-Embedded Servos** configuration.

5.8 Control Systems

The first step in controller design is to create a mathematical model of the system. The Ball on Plate system is described by state-space linear models for the X and Y axes. The state-space equations represent the dynamics of the system and the relationship between inputs, states, and outputs.

Open-loop control is a control strategy where the control inputs are predetermined without considering the system's feedback. In the context of the ball and plate system, open-loop control involves sending pre-defined commands to the servo motors without actively monitoring the ball's position or servo's angle. This approach assumes that the system's dynamics are well-understood and that external disturbances are negligible. Although it is easy to implement, it lacks adaptability hence may not be suitable for most scenarios.

The Proportional-Integral-Derivative (PID) Controller is a control strategy that operates based on three primary control actions: proportional, integral, and derivative. In the context of the ball and plate system, two independent classical single-loop PID controllers are used. One controller is used for the balls x-position and another controller is used for the balls y-position. The proportional action responds to the current error, the integral action addresses accumulated error over time, and the derivative action considers the rate of change of error [15].

The Model Predictive Control (MPC) controller design for the Ball on Plate system combines predictive control with optimization to achieve precise control of the ball's position. The controller uses a cost function to balance set-point tracking and control input smoothness. Additionally, the use of a control horizon and prediction horizon allows the algorithm to consider future behaviour and disturbances to make informed control decisions. The constraint handling ensures that the control inputs are physically feasible. The algorithm is run continuously to keep updating the instructions to the servo based on the current state [15].

Another study uses a two-DOF closed-loop approach using a Linear Quadratic controller. The controller consists of feed-back and feed-forward components, which continuously adjust control signals to minimize the difference between the desired ball position and its actual position. By iteratively correcting errors and considering the desired trajectory, the control system effectively stabilizes the ball at the centre of the plate and accurately responds to changes in the reference signal, enabling precise ball movement control. [8]

Table 12 compares the four controller Designs used for a Ball-in-Maze and Ball-on-Plate Model.

Table 12: Comparison of controller designs

Controller	Implementation Complexity	Feedback	Model Requirement	Adaptability	Precision	Resource Intensity
PID Controller	Low	Yes	Not required	Limited	Moderate	Low
MPC	Moderate	Yes	Required	High	High	High
Open-Loop Control	Low	No	Not required	None	Low	Low
Linear Quadratic Controller	Moderate	Yes	Required	Moderate	High	Moderate

Therefore, in the scope of this project, the PID controller will be used to obtain a good balance of precision and complexity.

5.9 Maze Solving Algorithms

The wall-following algorithm is a type of maze-solving algorithm where the ball follows the walls of the maze to navigate through it. The algorithm ensures that the ball always keeps one side (left or right) in contact with the wall while moving [19]. The advantage of this approach is its simplicity and easy implementation [19]. However, it may not always find the shortest path and can take a long time to reach the destination, especially in complex mazes.

The Modified Wall-Follower System (MWFS) is an efficient maze-solving algorithm that addresses issues faced by existing Wall Following Algorithms. It follows the Right-Left-Front hand rule for searching priority. By exploring right paths first, then left paths, and finally front paths, it ensures the solver never enters infinite loops and finds the destination point. MWFS can also explore all possible paths, enabling computation of the shortest paths between any two points in the maze as shown in **Figure 10**. This makes MWFS an effective and reliable maze-solving approach [20].

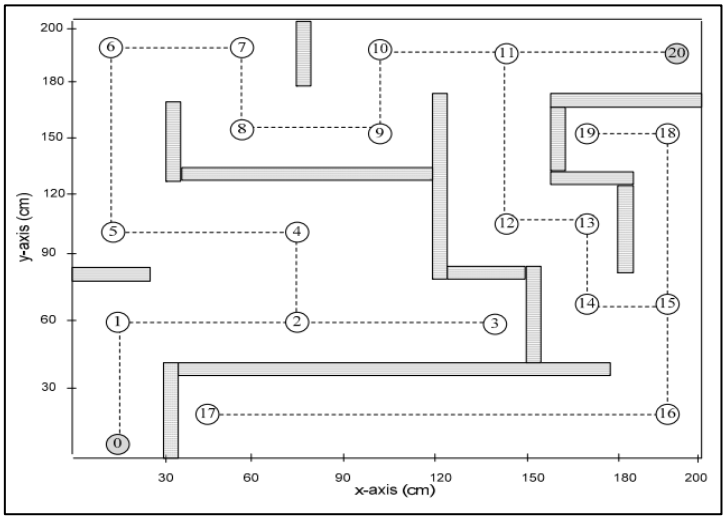


Figure 10: Path taken using Modified Wall-Follower System (MWFS).

Breadth-First Search (BFS) is a graph search algorithm used to find the shortest path from a start point to end point. It begins exploration at the start point and explores all its immediate neighbours before moving on to the next level neighbours. The Algorithm uses a cost storage to determine the order the cells have been explored. It explores layer by layer until finding the goal or end cell. By using this method, it is guaranteed that the shortest path will be found.

Unlike Breadth first search which explores neighbours in a breadth first manner, Best First Search chooses which neighbour to explore using an evaluation function. The evaluation function measures the current node to the goal.

A* is a widely used search algorithm for pathfinding especially for maze solving. It efficiently finds the path with the smallest cost from the start point to the end point in a graph or maze. It searches for shorter paths first and will consider all possible paths. A major drawback of the algorithm is its space and time complexity. It takes a large amount of space to store all possible paths and a lot of time to find them.

The flood-fill algorithm is a popular maze-solving approach based on determining the area in a multidimensional array connected to a given node in the maze [19]. The robot floods the maze by scanning each cell, marking its distance from the goal. It assigns scalar values to each cell, representing the distance from the ball to the final goal. The robot then selects the cell with the lowest score.

The modified flood-fill algorithm is an improvement over the basic flood-fill algorithm. It also assigns scalar values to each cell representing the distance from the robot to the final goal. However, MFFA updates cell

values using the recursive principle of bellmen, optimizing the objective value-function to maximize efficiency [20] . The MFFA guarantees both the shortest path and the fastest one to reach the final destination. It is considered one of the most effective maze-solving algorithms.

RRT (Rapidly Exploring Random Trees) is a sampling-based algorithm that rapidly explores the search space by randomly sampling states and connecting them via straight lines. It is scalable to high-dimensional spaces and expands quickly to cover wide areas in a short time [21].

On the other hand, RRT* (Rapidly Exploring Random Trees*) is an extension of RRT that guarantees convergence to the optimal path given enough time and samples. It continuously optimizes the tree structure, leading to near-optimal solutions as the explored region becomes denser [22]. Despite its high computational demands, RRT* is advantageous for tasks where optimality is crucial, and it excels in unbounded spaces.

Table 13: Comparison of Maze Solving Algorithms.

Algorithm	Guaranteed Path	Guaranteed Shortest Path	Memory Usage	Average Speed	Better Suit for Weighted Mazes
Wall Following	No	No	Low	Low	Non-weighted
Modified Wall Following	Yes	No	Moderate	Low	Non-weighted
Breadth First Search	Yes	Yes	Moderate	Moderate	Non-weighted
Depth First Search	Yes	No	Low	Moderate	Non-weighted
Best First Search	Yes	No	Moderate	Moderate	Non-weighted
A* Algorithm	Yes	Yes	High	High	Non-weighted
Dijkstra's Algorithm	Yes	Yes	High	High	weighted
Flood Fill Algorithm	Yes	Yes	Moderate	Moderate	Non-weighted
Modified Flood Fill	Yes	Yes	High	High	Non-weighted
RRT	Yes	No	High	High	Non-weighted
RRT*	Yes	Yes	High	High	Non-weighted

The comparative analysis (**see Table 13**) of maze-solving algorithms reveals varying strengths and suitability for different scenarios. It is essential to seriously consider these factors when choosing the algorithm for this project.

6 Requirements, Specifications and Approach

6.1 Requirements

The design requirements of this project will be broken down into four categories being the High-level requirements, financial requirements, Physical requirements, and Non-functional requirements. Michael Zenere, the project supervisor, has verified that all the requirements are feasible given the budget and timeline. There are no other stakeholders in this project.

The requirements for the project are highlighted in **Table 14**

Table 14: Project Requirements.

High level requirements	
[H.001]	Automatically detect the placement of a ball in the maze
[H.002]	Ensure precise navigation of the ball to the goal
[H.003]	Return to a neutral orientation upon task completion
[H.004]	Display relevant information on the screen during operation
[H.005]	Allow configurations (e.g. Goal Position) via the touch pane
Financial requirements	
[F.001]	Develop project within the budget of $150 of additional resources
Physical Requirements	
[P.001]	Ensure portability with a weight limit of 15kg
[P.002]	Maximum dimensions of 50cm x 50cm x 50cm (L x W x H)
[P.003]	The dimension of the square maze is to be 15cm x 15cm
[P.004]	The wall colour of maze is to be different that the base for easy image preprocessing
[P.005]	Operating temperature of 0°C to 50°C
Non-Functional Requirements	
[N.001]	Implement a user-friendly GUI and interface
[N.002]	Develop the main software in Python

6.2 Specifications and Approach

After comprehensive state of field, the decision on the approaches were made by considering several factors. **Figure 11** highlights the overview of the design and specifications.

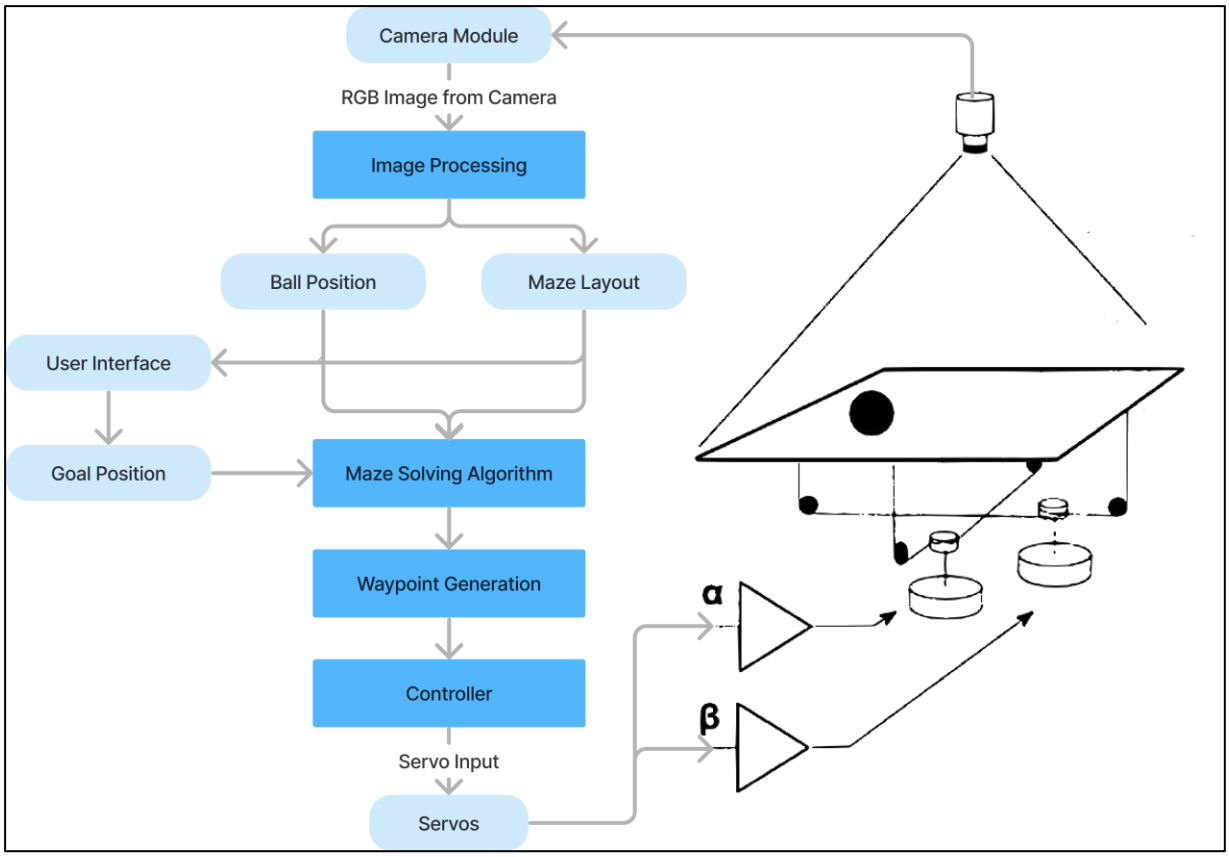


Figure 11: Design Overview of Ball in Maze Solving Robot.

Based on **Figure 11**, **Table 15** outlines the specifications, approach, and design choices for each aspect of the project.

Table 15: Project Specifications & Approach.

Aspect	Approach and Specifications	Design Choice and Justification
Camera Module	Utilize a camera module to capture RGB images of the maze environment, including the ball's position and maze layout. Choose based on resolution, frame rate, and compatibility	Raspberry Pi Camera Module chosen for its high-quality and high-speed image capture and compatibility with Raspberry Pi microcontroller.
Image Processing	Develop advanced algorithms for accurate ball detection and maze analysis. Optimize for real-time performance and computational efficiency.	Colour-based segmentation and adaptive thresholding for ball detection and maze layout detection.
Maze Solving Algorithm	Implement an efficient algorithm for determining the optimal path from current position to goal. Consider maze complexity and computational resources.	RRT* selected for high speed and path guaranteed.
User Interface	Design a user-friendly GUI in Python to visualize robot progress, display information, and allow user interactions such as configuring goal position via the interface.	Flask chosen for simplicity and ease of web-based interface creation.
Control System	Develop sophisticated control algorithms using servo motors for precise movement and	Raspberry Pi for vision processing and PSOC for efficient servo

	positioning of the ball within the maze. Aim for high accuracy in navigation.	communication. PID control for controlling the ball's movement in the maze.
Servo	High-speed, high-accuracy servos to control the tilting platform precisely and quickly.	Dynamixel AX-12 chosen for its high-speed, high-accuracy performance. Important to note that supervisor Michael Zenere has prior experience working with these servos which will be helpful.
Safety Features	Incorporate safety mechanisms (e.g. hard limits and emergency stop) to prevent accidents. Ensure compliance with safety testing standards.	Angle limiting, emergency stop button, and ensuring the robot will not run when the ball isn't present.
Cost-effectiveness	Optimize component selection and design to maintain project within budget of $150. Focus on affordability while meeting functional requirements.	University facilities used for cost-effective manufacturing including laser cutting and 3d printing services. Software-heavy design reduces hardware complexity and costs.

The decisions outlined above ensure that the project remains within the budget while meeting all functional requirements.

7 Results and Discussion

7.1 Summary of work completed.

This section provides a detailed account of the work completed throughout the project, presented in chronological order. For precise project timelines, please refer to the Gant chart in **Section 9.2**.

7.1.1 Research and Planning

In the early stages of the project, thorough research and planning has been conducted. This process is important as it avoids wasting time, unnecessary expenses, and the need for redesigns.

First an in-depth state of field analysis is conducted to gain valuable insights into existing technologies and methodologies (**see Section 5**). Thorough consultation with the stakeholder, Mr. Michael Zenere, the project supervisor, was undertaken to determine the realistic and achievable Functional, Non-Functional and Stretch project requirements (**see Section 6.1**).

Once requirements are identified, it is evaluated to make informed decisions about design specifications and approach (**see Section 6.2**). This approach establishes a solid foundation for project execution and aids in achieving our goals and objectives.

7.1.2 Maze Puzzle Design

Before designing the system, the specification of the Maze Puzzle design and construction had to be finalised. This ensures that the system can be successfully built around the specified maze design. **Table 16** highlights the key considerations for the design.

Table 16: Maze Puzzle Design Considerations.

Key Consideration	Approach
Interchangeable Mazes	The system is designed to accept any maze that meets the specified dimensions allowing for interchangeable mazes.
Standalone Playability	The mazes are also designed as standalone games, allowing for manual play.
Dimensions and Construction	The mazes are 15 cm by 15 cm squares made up of 2 layers of 3mm coloured acrylic sheet. This size is convenient for handheld games. The bottom layer is white, and the top layer is red to simplify colour segmentation.
Ball Selection	A 9mm black ceramic ball bearing is used. This size is cost-effective and suitable for the maze of this size. It is important to note, that the system is designed to detect and control all ball size that can fit on the Maze, but varying the ball size will require retuning of the PID controller. Hence, for the current scope of the project, a constant ball size will be used.

Figure 12 shows a sample of the constructed maze design.

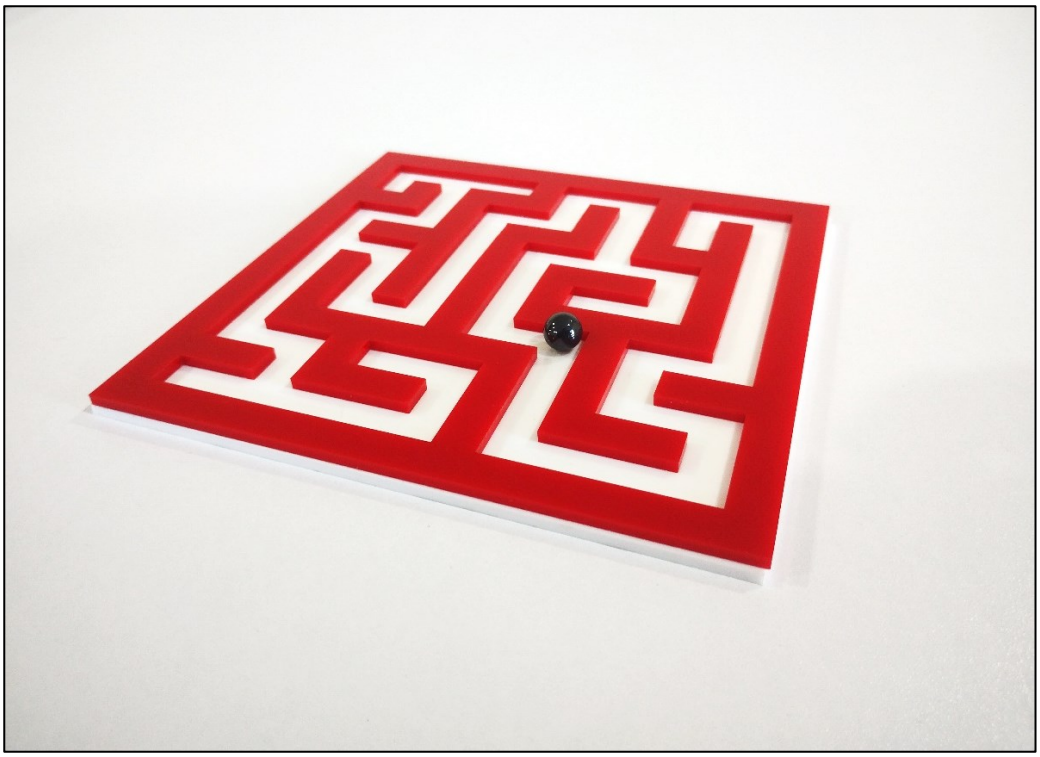


Figure 12: Constructed Maze Design.

7.1.3 Tilting Platform Design

The design choice is a two-degree-of-freedom (2-DOF) tilting platform that allows controlled tilting along both the x and y axes. The servos are embedded in the frame similar to a 2-axis gimbal. The completed assembly of the structure is shown in the **Figure 13**. Refer to **Appendix G** for the CAD drawings of the assembly and each component.

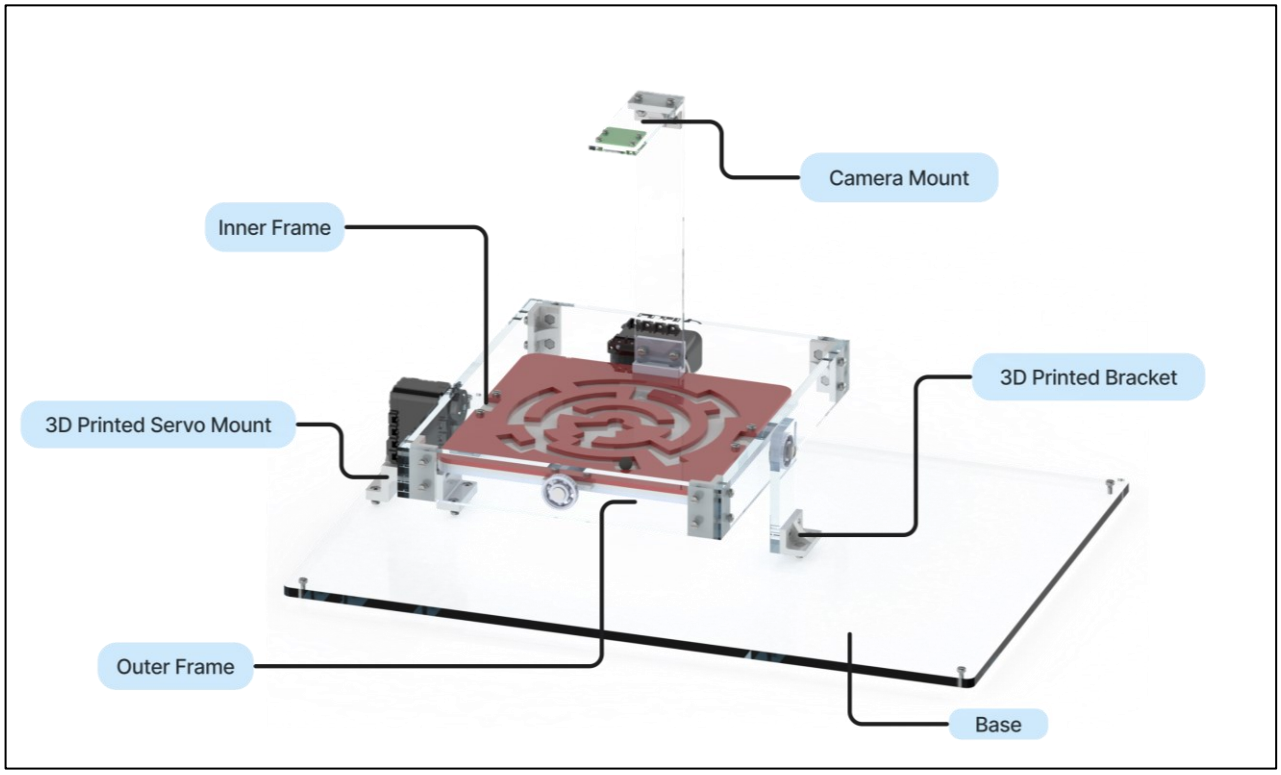


Figure 13: Assembly of Ball in Maze Solving Robot.

Table 17 explains the construction and functionality of the components of the tilting platform.

Table 17: Construction And Functionality of the Components of the Tilting Platform.

Component	Construction, Functionality and Features
Base	Constructed from 6mm clear acrylic sheet. It has four corner holes with rubber feet for stability, and additional holes for servo mounts and brackets, all sized for M3 screws.
Servo Mount	Printed in ABS and attached to the base via M3 10mm screws and nuts. It features hexagon countersinks to lock the nut in place, ensuring a flush surface.
Brackets	Printed in ABS, these brackets serve as 90-degree joints for the supports, outer frame, inner frame, and camera mount. It also has hexagon countersinks to lock the nut in place, ensuring a flush surface.
Outer Frame	Constructed from four 6mm clear acrylic sheets, assembled with four brackets. Supported by the outer servo on one end and a bearing on the other, it tilts along the x-axis and supports the inner frame inside.
Inner Frame	Constructed from three stacked 3mm acrylic sheets. Attached to the inner servo on one end and to a shaft on a free-rotating bearing at the other end. The top acrylic layer acts as a border around the platform to snugly fit the maze, with openings on either side for easy removal or replacement.
Camera Mount	Constructed from two 3mm clear acrylic sheets mounted at 90 degrees via a bracket. The top plate has an opening with M2 holes to mount the camera facing down on the maze. It is directly connected to the inner frame, ensuring it remains perpendicular to the tilting platform, simplifying image processing. The total height from the Maze to the top of the camera mount is 16.5 cm, within the required height limit.
Miscellaneous	- M3 and M2 bolts and nuts for all joints and connections - Bearings for smooth rotation along the x and y axes - Rubber feet to support the base

7.1.4 Electrical Circuit Design

Based on the design of the Tilting platform, the circuit diagram of the system was designed. The circuit diagram displays the interconnection of various electrical components within the maze-solving robot's system. It illustrates the power supply, signal paths, and connections between different input and output devices, providing an overview of the electrical layout as shown in **Figure 14**. The system works by using a Raspberry Pi for high-level decision making and computer vision, and a PSOC for precise servo control, to tilt the platform and guide the ball through the maze based on visual feedback.

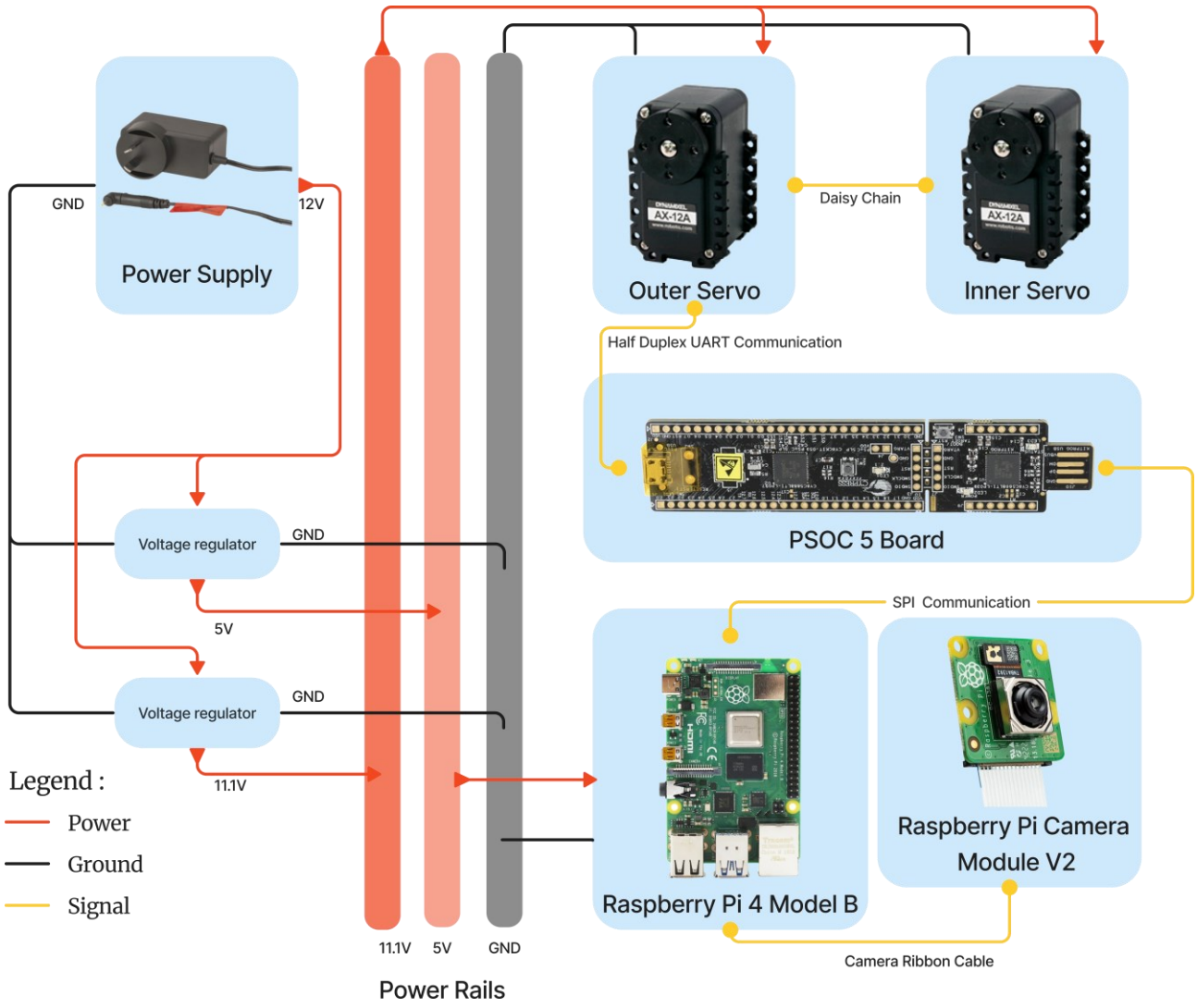


Figure 14: Circuit Diagram Overview

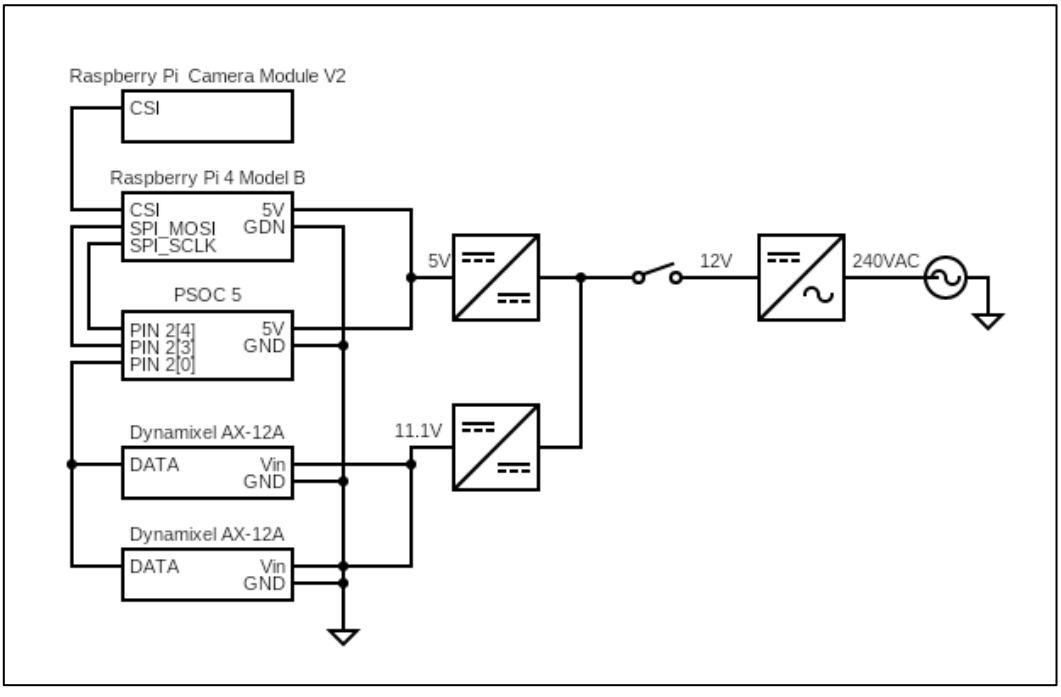


Figure 15: Circuit Design of System

Figure 15 shows the technical circuit and highlights all connections.

The power supply for the system is specified as 12VDC, 5A, 65W using a power adapter. This is justified based on the power consumption of the components: the Dynamixel servos, each drawing a maximum of 1.5A at 11.1V, and the Raspberry Pi, which draws up to 3A at 5V, with the PSOC drawing negligible current. The total current draw is calculated as follows.

$$Current\ Required\ at\ 12V = 2 * \left(\frac{11.1\ V}{12\ V}\right) * 1.5A + \left(\frac{5\ V}{12\ V}\right) * 3A + I_{psoc} \approx 4.1\ A \leq 5A$$

This results in approximately 4.1A, which is well within the 5A capacity. Voltage regulators step down the 12V supply to 11.1V for the servos and 5V for the Raspberry Pi and PSOC, ensuring optimal operating voltages for all components.

The actuators used for the 2-degree-of-freedom (2-DOF) tilting platform are the Dynamixel 12AX servos. These servos are integral to the maze control system, offering high accuracy and precision. Key specifications are highlighted in **Table 18**.

Table 18: Key Specifications of the Actuators.

Specification	Details and Justification
Power Supply	11.1V which matches the recommended voltage requirement for the Dynamixel 12AX servos.
Position Control	High resolution of 1024 steps, translating to 0.29° per step. This ensures precise tilting of the platform, which allows for minute adjustments when guiding the ball through the maze.
Speed	The no-load speed of 59 revolutions per minute at 12V ensures responsive platform adjustments for maze navigation. Given the system's design limits of ±20 degrees, the slowest transition time from -20 to 20 degrees is calculated as: $T = (20° - (-20°)) * \frac{60\ s}{59*360°} = 0.11\ s$ This speed is more than sufficient for the required movements.
Torque	Stall torque of 1.5 N.m at 12V and 1.5A. This offers adequate force to move the platform reliably under load conditions, maintaining control and stability.
Control Capabilities	Position control for precise platform angles, and velocity control for smooth motion. Enables the system to target specific angles and achieve controlled tilting speeds, ensuring smooth ball movement.
Communication	Half-duplex UART communication with the PSOC microcontroller.
Configuration	One servo for the inner frame and one for the outer frame

For the main controller of the system, a combination of the PSOC controller and a Raspberry Pi 4 Model B is used. Both of these microcontrollers are excellent yet built for different purposes. **Table 19** highlights the comparison of the capabilities of each microcontroller for various task relevant to this system.

Table 19: Comparison of the Capabilities of each Microcontroller.

Task	Raspberry Pi	PSOC
Computer Vision (Ball and Maze detection)	Good	Bad
Wi-Fi (Web-based User Interface)	Good	Bad (Requires External Module)
General Input Output	OK	Very Good
Serial Communications (Communication with Servo)	Ok	Very Good

As seen in **Table 19**, neither one microcontroller is the best choice for this system. Furthermore, even if a single microcontroller can perform all the required operations, the processing power may still be a concern.

As such, the decision was made to distribute the task across the two microcontrollers. The Raspberry Pi will handle most of the high-level decisions and computer vision tasks while the PSOC will handle the servo control. This design is accomplished via a Master-Slave communication protocol via the SPI ports.

For the vision-based task, the Raspberry Pi Camera Module V2 is used. The specifications of the camera are highlighted in **Table 20**.

Table 20: Camera Specifications.

Specification	Details
Weight	3g
Still Resolution	8 Megapixels
Horizontal Field of View (FoV)	62.2 degrees
Vertical Field of View (FoV)	48.8 degrees

From the **Table 20**, we can see that the camera is light enough to be supported by the camera mount without any flexing. The camera provides high resolution for image processing. The height of the camera mounted from the maze is calculated as follows.

$$H = \frac{15\ cm * \frac{1}{2}}{\tan\left(\min(62.2°, 48.8°) * \frac{1}{2}\right)} = 16.5\ cm$$

As shown in **Figure 16**, communication between the camera and the Raspberry Pi is facilitated through the Camera Serial Interface (CSI).

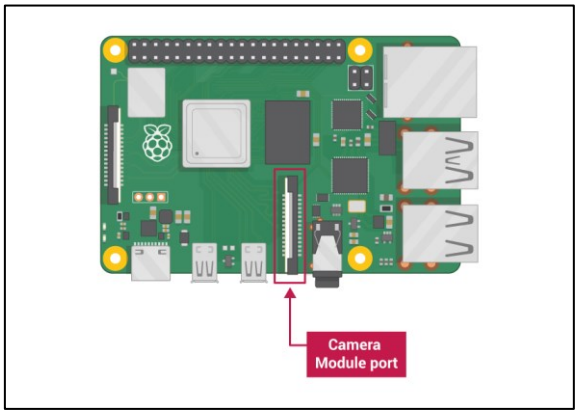


Figure 16: Camera Serial Interface (CSI).

7.1.5 Communication Protocols Raspberry Pi and PSOC

The communication protocol used to communicate between the Raspberry Pi and PSOC is a Master Slave SPI protocol. The SPI is well suited for high-speed communications over 20 Mbps. It is simple in design requiring just 3 wires and a ground.

1. MOSI Master out slave in
2. MISO Master in slave out
3. Clock
4. Ground

In the scope of this project, the Raspberry Pi acts as the Master and the PSOC acts as the Slave. The data required to be sent is the target position of Servo 1 and Servo 2. Since the PSOC doesn't need to send any information back to the Raspberry Pi, the MISO connection can be excluded. **Table 21** and **Figure 17** shows the pins connections used for the SPI Communication.

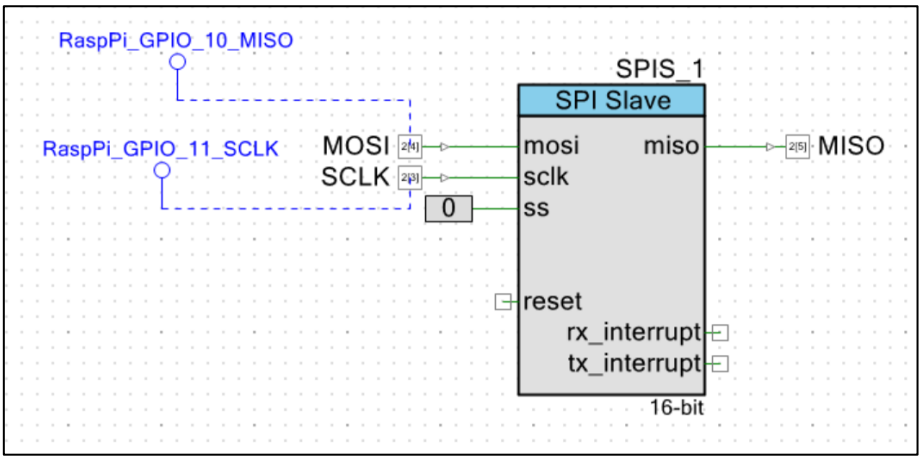


Figure 17: SPI Communication Schematic.

Table 21: SPI Pin Connections.

Wire	Raspberry Pi 4	PSOC 5
Clock	GPIO 11 (SCLK)	Pin 2[4]
MOSI	GPIO 10 (MOSI)	Pin 2[3]
Ground	GND	GND

An important note to consider is that the Raspberry Pi operates at 3.3V, so the output voltage of the PSOC had to be adjusted from its default 5V to match. PSOC 5 features multiple VDDIO pins, each powering different ports as shown in **Table 22**.

Table 22: PSOC VDDIO Pins.

VDDIO	Port Pins
VDDIO0	P0[7:0], P4[7:0], P12[3:2]
VDDIO1	P1[7:0], P5[7:0], P12[7:6]
VDDIO2	P2[7:0], P6[7:0], P12[5:4], P15[5:4]
VDDIO3	P3[7:0], P12[1:0], P15[3:0]
VDDD	P15[7:6] (USB D+, D-)

As such the value of VDDIO2 was set to 3.3V to match the raspberry Pi. The data being sent is the target position of each servo relative to its neutral position in steps (0 – 1024). A limit of 50 steps in either axis is set to prevent the platform from colliding with the base of the robot. The implementation involves a simple data transfer of 16 bits which is shown in **Figure 18** and **Table 23**.

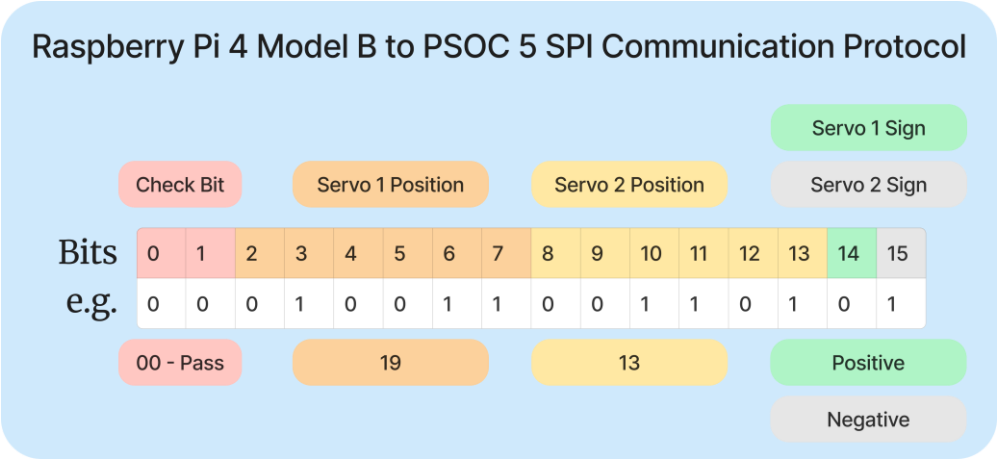


Figure 18: SPI Communication Protocol.

Table 23: SPI Communication Protocol.

Bits	Significance
0-1	0b00 (Check Bit)
2-7	5-bit Integer (Servo 1 Position)
8-13	5-bit Integer (Servo 2 Position)
14	0 for positive, 1 for negative (Servo 1 Sign)
15	0 for positive, 1 for negative (Servo 2 Sign)

Bits 0-1 act as a check bit (0b00), while bits 2-7 and 8-13 represent 5-bit integers for Servo 1 and Servo 2 positions, respectively. Bit 14 indicates the sign for Servo 1 (0 for positive, 1 for negative), and Bit 15 indicates the sign for Servo 2 in a similar manner. The data is transferred via two shift registers, in the Raspberry Pi and the PSOC 5 respectively. On every Clock pulse, 1 bit is shifted from the master to the slave, and this is repeatable until all the 16 bits have been sent. The bit rate used is 33 Mbps.

7.1.6 Communication Protocols PSOC and Dynamixel

The PSOC connects to the Dynamixel servos through a TTL-level multi-drop bus using a half-duplex UART interface. The pin connections are as shown in **Table 24**.

Table 24: Half Duplex UART connection of PSOC and Dynamixel.

Wire	Raspberry Pi 4
Ground (GND)	Common Ground Reference
Power (VDD	11.1V input from Power Rail
Data (DATA)	Pin 2[0] PSOC 5

To achieve half-duplex UART communication, the DATA pin functions as both a transmitter and receiver, sending and receiving packets to and from the servos. This is implemented by setting the pin as both a digital input as well as digital output as shown in **Figure 19**.

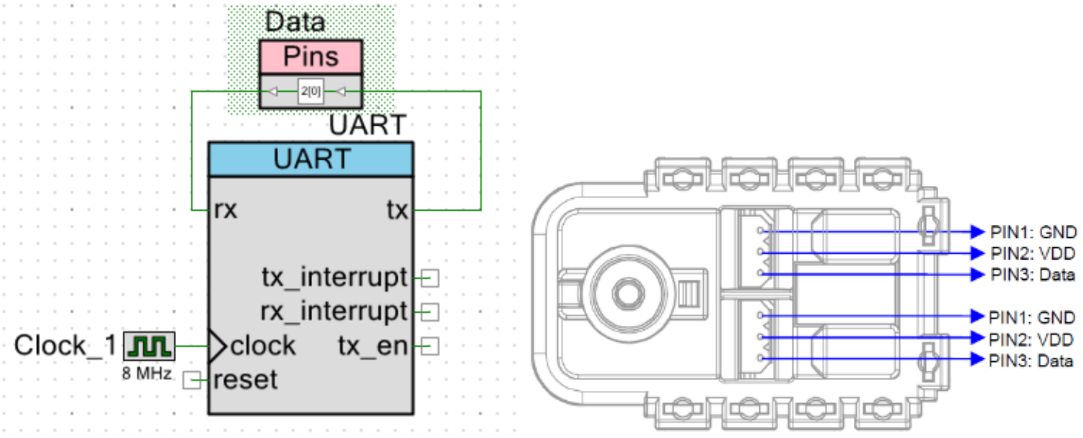


Figure 19: Half Duplex UART connection of PSOC and Dynamixel.

The communication protocol implemented in the PSOC for controlling the Dynamixel servo motors involves a series of predefined instructions and packet structures. These instructions are used to send commands and read responses from the servos. **Table 25** summaries the functions implemented for controlling the servo.

Table 25: Dynamixel Communication Protocol.

Function	Description	Packet Structure
LED_Control	Control the LED on the servo.	`[0xFF, 0xFF, ID, 4, 3, 25, Status, Checksum]`
Set_ID	Set a new ID for the servo.	`[0xFF, 0xFF, ID, 4, 3, 3, new_ID, Checksum]`
Move	Move the servo to a specified position.	`[0xFF, 0xFF, ID, 5, 3, 30, Position_L, Position_H, Checksum]`
MoveSpeed	Move the servo to a position at a specified speed.	`[0xFF, 0xFF, ID, 7, 3, 30, Position_L, Position_H, Speed_L, Speed_H, Checksum]`
TorqueStatus	Enable or disable the torque.	`[0xFF, 0xFF, ID, 4, 3, 24, Status, Checksum]`
SetReturnLevel	Set the status return level.	`[0xFF, 0xFF, ID, 4, 3, 16, level, Checksum]`
reset	Reset the servo to factory settings.	`[0xFF, 0xFF, ID, 2, 6, Checksum]`
ping	Ping the servo to check if it's online.	`[0xFF, 0xFF, ID, 2, 1, Checksum]`
MoveSpeedRW	Register a move command with speed for delayed execution.	`[0xFF, 0xFF, ID, 7, 4, 30, Position_L, Position_H, Speed_L, Speed_H, Checksum]`
action	Execute the registered move command.	`[0xFF, 0xFF, 254, 2, 5, 250]`
ReadVoltage	Read the current voltage from the servo.	`[0xFF, 0xFF, ID, 4, 2, 42, 1, Checksum]`
ReadTemperature	Read the current temperature from the servo.	`[0xFF, 0xFF, ID, 4, 2, 43, 1, Checksum]`
ReadPosition	Read the current position from the servo.	`[0xFF, 0xFF, ID, 4, 2, 36, 2, Checksum]`
ReadSpeed	Read the current speed from the servo.	`[0xFF, 0xFF, ID, 4, 2, 38, 2, Checksum]`

The full code for each function can be found in **Appendix F**. The Packet fields include:

- ID: Identifier of the servo (0-253).
- Length: Length of the instruction packet.
- Instruction: Command to be executed (e.g., AX_WRITE_DATA, AX_READ_DATA).
- Parameters: Data specific to the instruction (e.g., position, speed).
- Checksum: Error-checking byte.

During integration with the entire system, the main function sets up UART and SPI communication, and controls the servos based on received SPI data. It continuously reads data from the SPI interface, processes it, and sends commands to the servos accordingly. Since the commands require an ID for each servo, the servos can be wired in daisy chain to simplify the wiring.

7.1.7 Ball Localisation and Tracking

The system utilizes computer vision techniques on the image captured by the Raspberry Pi Camera to detect the ball position as shown in **Figure 20**.

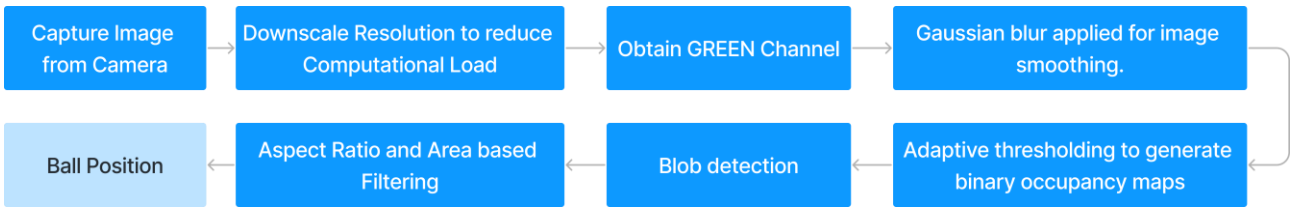


Figure 20: Flowchart of Ball Detection process.

The image resolution is reduced to 200x200 pixels to increase capturing speed, averaging about 1 ms. This is adequate given the maze size of 15x15 cm and the ball size of 9 mm. No additional preprocessing is required since the camera is always perpendicular to the platform.

The ball is black, the floor is white, and the walls are red. Hence, the blue or green channel can be used to isolate the ball, avoiding the red channel to exclude the walls. The image undergoes Gaussian blur for smoothing and adaptive thresholding to generate a binary occupancy map.

Blob detection with 4-connectivity is performed. Next, aspect ratio and area-based filtering is performed to remove any falsely detected artifacts (if present). Finaly, the centre of the detected blob represents the ball's position, with the entire process taking less than 2 ms.

Figure 21 illustrates the image captured by the camera, the extracted green channel, the smoothed green channel, and the binary occupancy map of the ball. For the full code, refer to **Appendix E**.

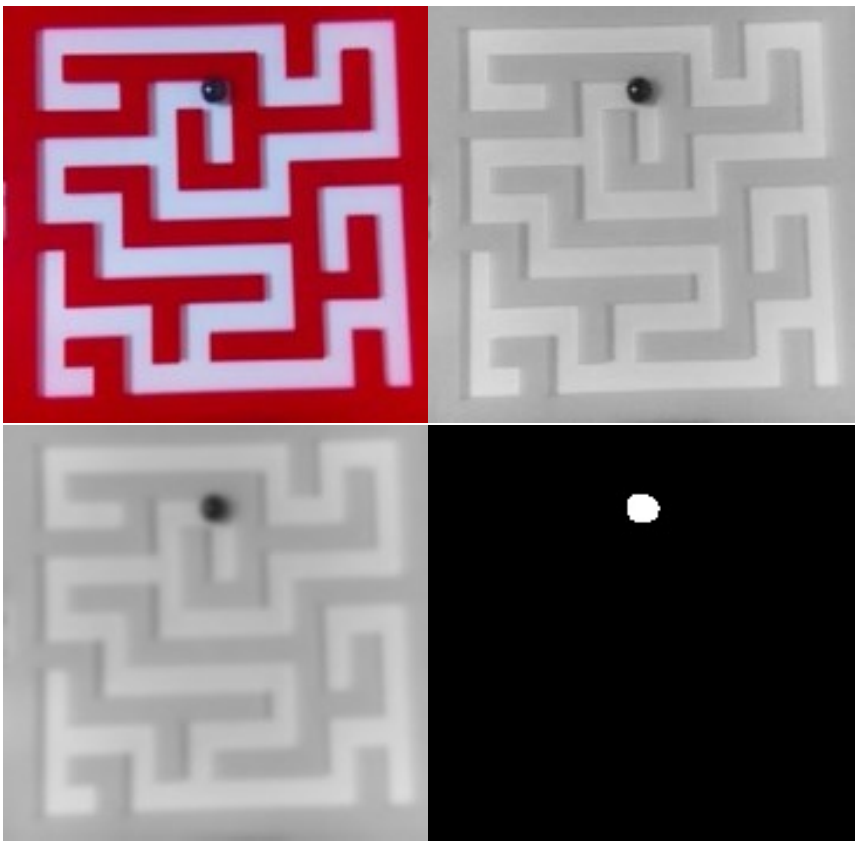


Figure 21: Ball Detection Images

7.1.8 Maze Layout Detection

Figure 22 shows the flowchart of the maze layout detection process.

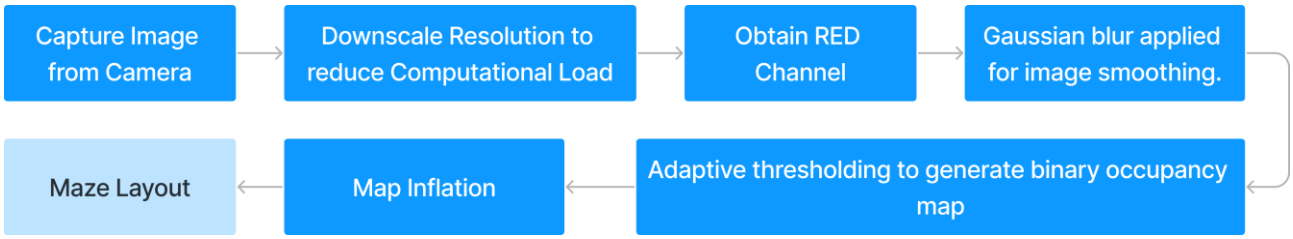


Figure 22: Flowchart of the Maze Layout Detection process.

The system begins by capturing an image from the Raspberry Pi Camera with downscaled resolution. Since the maze walls are red, the RED channel of the image is extracted. This isolates the walls from the rest of the scene.

Then a Gaussian blur is applied to the extracted RED channel to smoothen the image, reducing noise and minor variations. Next, adaptive thresholding is applied to the smoothed RED channel to generate a binary occupancy map. This process separates the maze walls (binary high) from the rest of the scene (binary low). Finaly, map Inflation applied to act as a buffer region between the ball and obstacles to increase the reliability that the path is safe and valid. The resulting binary occupancy map provides a clear distinction between the maze walls and the open paths, allowing for accurate navigation and control within the maze.

Figure 23 illustrates the image captured by the camera, the extracted red channel, the smoothed red channel, the binary occupancy map of the maze layout, and the inflated binary occupancy map of the maze layout. For the full code, refer to **Appendix E**.

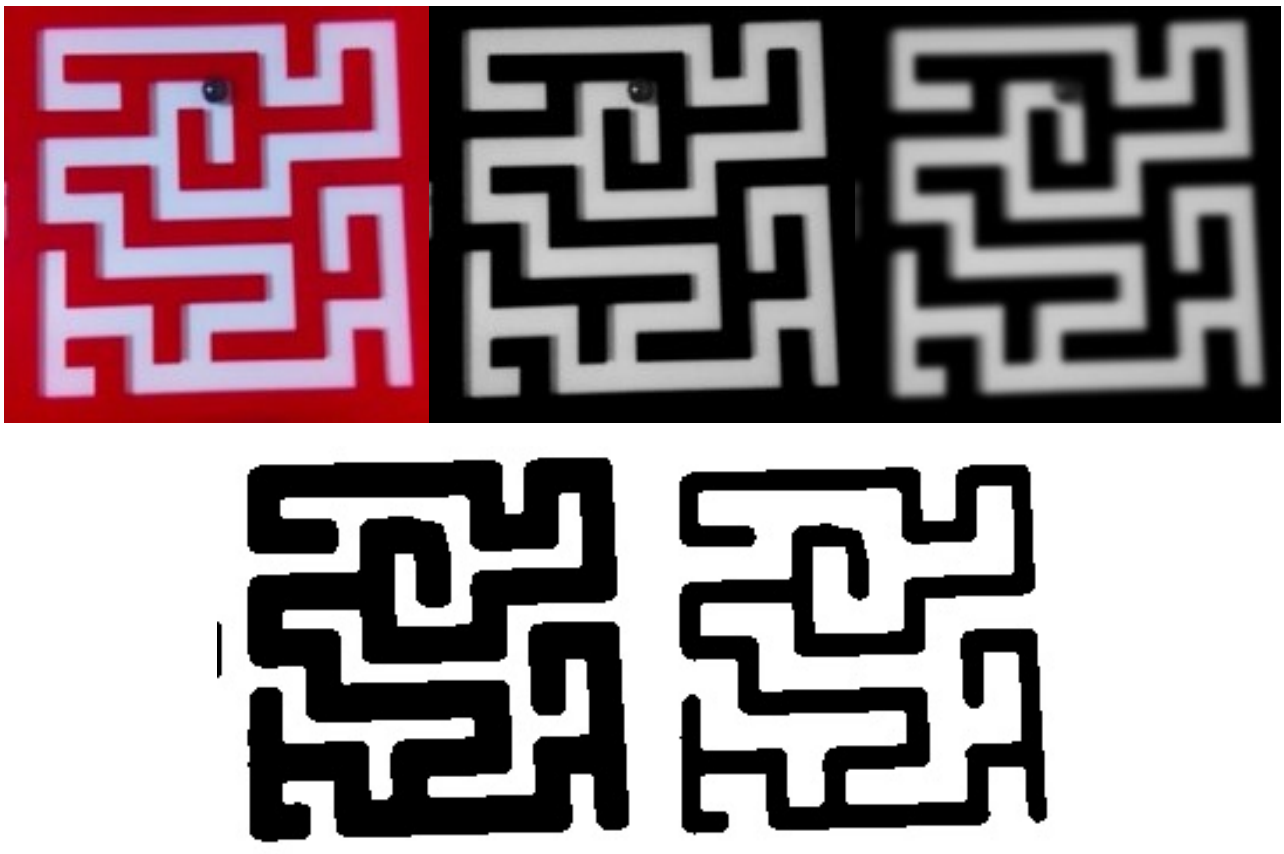


Figure 23: Maze Layout Detection Images

7.1.9 Path Planning

Figure 24 shows the Flowchart of Path Planning Process.

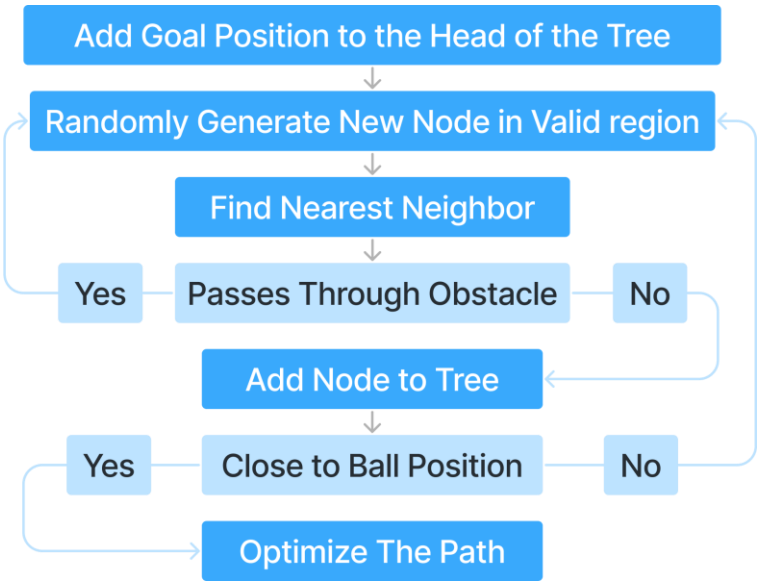


Figure 24: Flowchart of Path Planning Process.

1. The goal point was added to the head of the tree, initializing the RRT* algorithm with the goal position as the first node. After initialisation, obstacles within a set radius around the ball position are removed. This prevents the ball from being mistakenly considered an obstacle during path planning.
2. A new node is then randomly generated within the valid region of the occupancy map. The nearest neighbour to this newly generated node is identified within the tree. The algorithm then checks if a

direct path from the nearest node to the new node intersects any walls. If the path passes through an obstacle, the new node is discarded and the process of generating a new node is repeated.

3. If the path is obstacle-free, the new node is added to the tree and connected to the nearest node. The algorithm checks if the new node is close to the ball's current position. If the node is near the ball, the next step is to optimize the path. Otherwise, the algorithm continues generating new nodes.
4. Path optimization involves re-evaluating the parent nodes of each node within a specified radius to minimize the overall path cost. The result is an optimized path from the start position to the goal position, which avoids obstacles and minimizes the path length.

Figure 25 illustrates the inflated binary occupancy map of the maze layout, the RRT* node tree and the optimised final path. For the full code, refer to **Appendix E**.

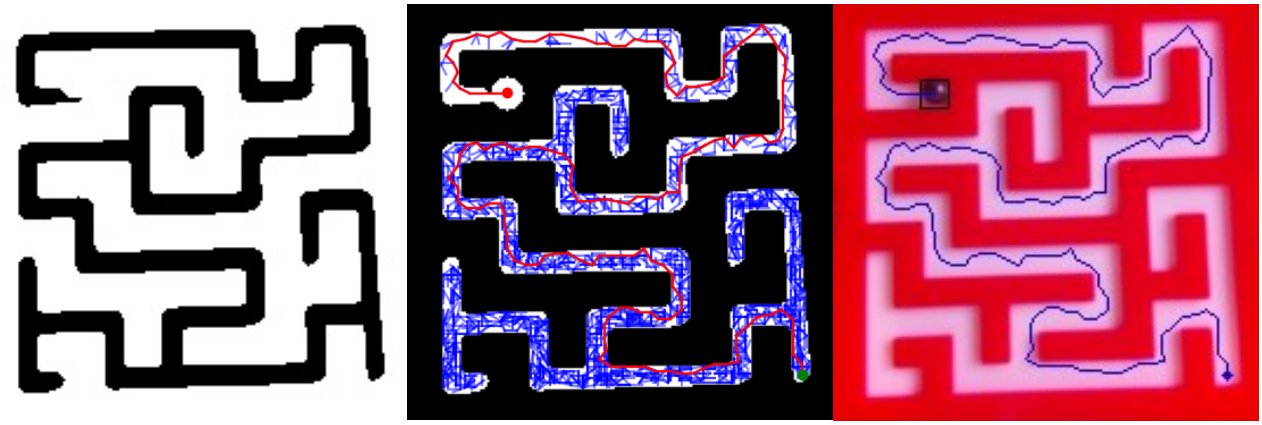


Figure 25: Path Planning (RRT) Images*

The number of Iterations, N, determines how many iterations the RRT* algorithm will perform during its execution. A higher value of N signifies more iterations, allowing for greater exploration of the search space. However, this also results in higher computational time due to the increased number of node expansions and path optimizations.

The “Stop When Reached” flag is a feature implemented to halt as soon as a valid path to the goal is found. If the algorithm successfully finds a path, it terminates immediately, regardless of the total number of iterations specified.

This approach offers several benefits. By stopping when a path is found, the algorithm saves computation time, particularly for straightforward paths. Users experience shorter wait times, enhancing usability. Despite stopping early for simple paths, the algorithm remains capable of finding more complicated routes ensuring it can handle diverse path requirements.

The goal position is set as the head of the tree instead of the ball’s position to ensure that the tree converges on the goal. This allows for recalculating the path to the same goal position using the existing tree if the ball deviates from its preplanned path. Furthermore, the RRT* algorithm, with its inherent probabilistic completeness, guarantees that it will find a path to the goal given sufficient time and a feasible path. This property assures users that, with appropriate settings, the algorithm can reliably navigate various environments and obstacles.

7.1.10 Control Systems

The overview of the control loop to solve the maze is highlighted in the following steps. For full code refer to **Appendix E**.

1. The system obtains the series of waypoints from the planned path. The initial waypoint is set as the target ball position.
2. Ball position is detected. If the ball is not detected, the system returns to a neutral state and waits. This feature ensures safety and allows for dynamic adjustments to the ball's position without disrupting the system. The system quickly recalculates the route using the existing tree when the ball is detected again.
3. Using the balls position, the system calculates the error between the current position and the target waypoint.
    a. If the error is below a threshold, the system shifts to the next waypoint.
    b. If the current waypoint is the last one, the system slowly returns the platform to a neutral flat position, indicating the goal is reached.
    c. In rare cases where the error exceeds a predefined threshold, indicating a significant deviation from the path, the system uses the existing tree to recalculate the path. This process is very fast since it utilizes the existing tree structure and nodes, ensuring efficient path correction. Waypoints are updated based on the recalculated path.
4. Based on the error calculated, a PID (Proportional-Integral-Derivative) control is used to determine the target servo position for both X and Y axes independently. The target servo position is then sent to the PSOC via SPI communication to control the servos.
5. This loop is repeated until the ball reaches the goal position or is cancelled by the User via the user interface.

Figure 26 displays the PID signals, including the target signal and output, illustrating how the system adjusts the servo position based on feedback.

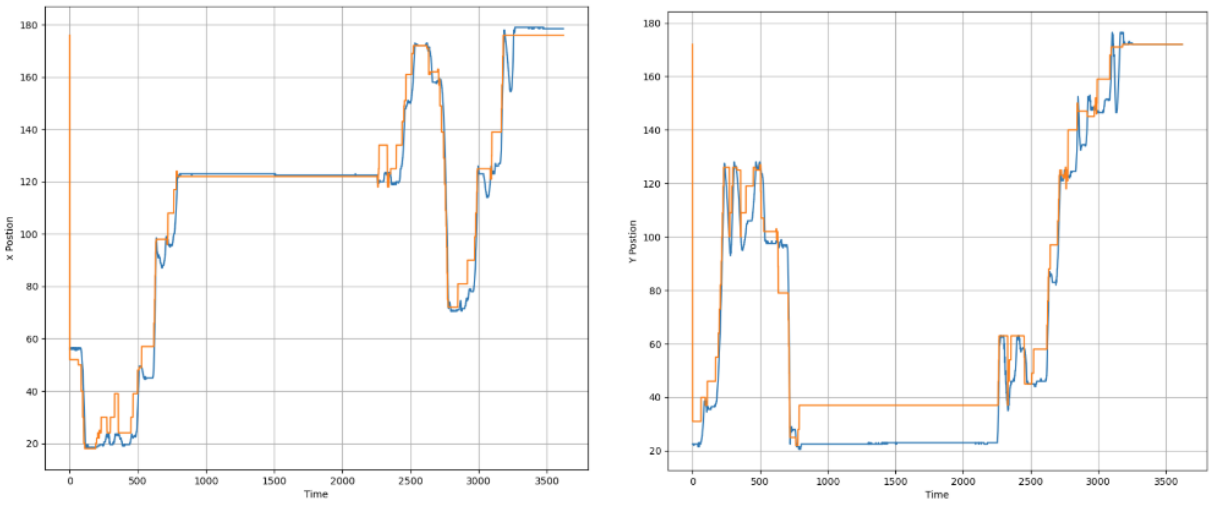


Figure 26: PID Signal for X position (Left) and Y position (Right).

7.1.11 User Interface

The graphical user interface is created as a Web-Based Interface using Flask, a Python web framework, along with HTML, CSS, and JavaScript [23] . Being web-based, the system can be accessed from devices connected to the same Wi-Fi network, such as phones, laptops or any external device connected to Wi-Fi. The interface is designed to be intuitive and user-friendly, enhancing the user experience as shown in the **Figure 27**. For full code refer **Appendix E**.

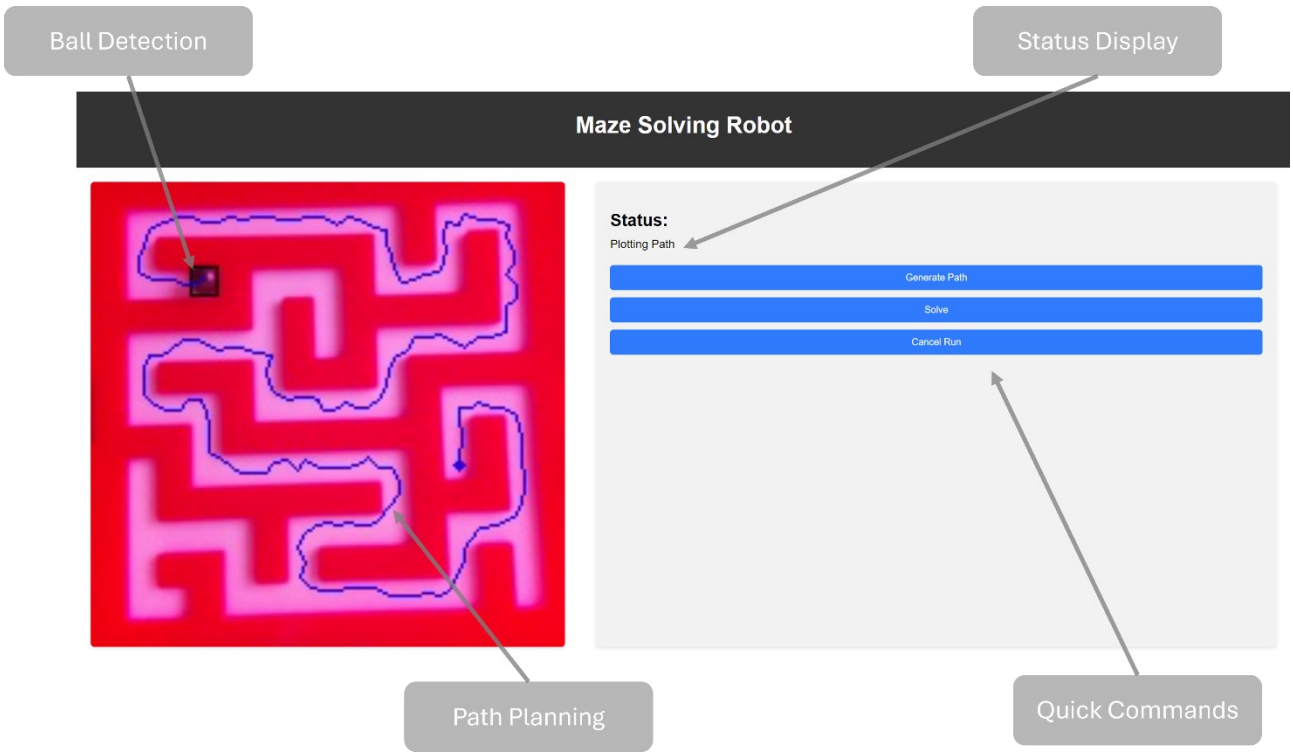


Figure 27: Web-Based Interface .

The features and functionality of the user interface are highlighted in the **Table 26**.

Table 26: User Interface Features.

Feature		Functionality
Live Video Feed		Provides a live video feed from the camera at 30 frames per second (fps) for real-time monitoring. Users can click anywhere on the screen to select the goal position, ensuring ease of interaction.
Status Display	Ready	Indicates that the system is ready and waiting for user input.
	No Ball Detected	Alerts the user when no ball is detected in the maze. This status ensures that the system does not proceed without the required input.
	Generating Path	The system is calculating the optimal path based on the selected goal position.
	Plotting Path	Displays the calculated path overlaying the live video feed, providing users with a visual representation of the planned route.
	Solving Maze	The system is actively controlling the ball to navigate through the maze towards the goal position. Users can observe the live movement of the ball as it progresses towards the goal.
	Reached Goal	The ball successfully reaches the goal position, marking the completion of the maze-solving task.
Action Buttons	Generate Path	Generates the optimal path based on the selected goal position using the RRT* algorithm.
	Solve	Begins solving the maze by controlling the ball to navigate towards the goal position.
	Cancel Run	Allows the user to cancel the current operation or run, providing an option to stop the maze-solving process if needed.

7.2 Findings

7.2.1 Final Bill of Materials (BOM)

The BOM is essential for project management, budgeting, and procurement. It provides a clear overview of the project's material and cost requirements. Most components are sourced from the same supplier, simplifying logistics. The system was designed to be more software driven to reduce the number of physical parts needed, hence lowering the overall cost.

The BOM includes all materials, components, and resources needed for the "Ball-in-Maze Solving Robot," detailing each item's source, product ID, unit, quantity, unit cost, and total cost as shown in **Table 27**.

Table 27: Bill of Materials (BOM).

No	Part Name	Source	Product ID	Unit	Quantity	Unit cost ($AUD)	Cost ($AUD)
1	Arduino Compatible DC Voltage Regulator Module	Jaycar Electronics	XC4514	each	2	10.25	20.5
2	2.1mm DC Plug with Screw Terminals	Jaycar Electronics	PA3711	each	1	7.95	7.95
3	12VDC 5A 65W Switch mode Mains Adaptor with 7 Changeable Plugs	Jaycar Electronics	MP3560	each	1	69.95	69.95
4	M3 x 10mm Steel Screws - Pack of 25	Jaycar Electronics	HP0403	each	1	3.55	3.55
5	M3 x 6mm Steel Screws - Pack of 25	Jaycar Electronics	HP0400	each	1	3.15	3.15
6	M3 Steel Nuts - Pack of 25	Jaycar Electronics	HP0425	each	1	3.70	3.70
7	608RS Carbon Steel Bearing 10 Pack	Jaycar Electronics	YG2950	each	1	13.95	13.95
8	Si3N4 Ceramic Ball GR.5 Black Colour, 9mm	AU Miniature Bearings	389530507	each	1	3.8	3.8
9	Raspberry Pi 4 Model B Board	Monash	-	each	1	0	0
10	PSoC™ 5LP CY8CKIT-059	Monash	-	each	1	0	0
11	Raspberry Pi Camera Module V2 (Including Mounting Screws)	Monash	-	each	1	0	0
12	Dynamixel AX-12 Servo (Including Mounting Screws)	Monash	-	each	2	0	0
13	Wires and Connectors	Monash	-	-	-	0	0
14	3mm Clear Acrylic sheet	Monash	-	m^2	0.08	0	0
15	3mm White Acrylic sheet	Monash	-	m^2	0.0225	0	0
16	3mm Red Acrylic sheet	Monash	-	m^2	0.135	0	0
17	6mm Clear Acrylic sheet	Monash	-	m^2	0.2	0	0
18	ABS (white)	Monash	-	g	50	0	0
						Total	**126.55**

Therefore, the total cost of the project, at **126.55 AUD**, is well within the budget of 150 AUD. Most components are sourced from the same supplier, which simplifies procurement. Additionally, the project utilizes university facilities such as 3D printing and laser cutting to save costs.

7.2.2 Final Product

Figure 28 depicts the final product.

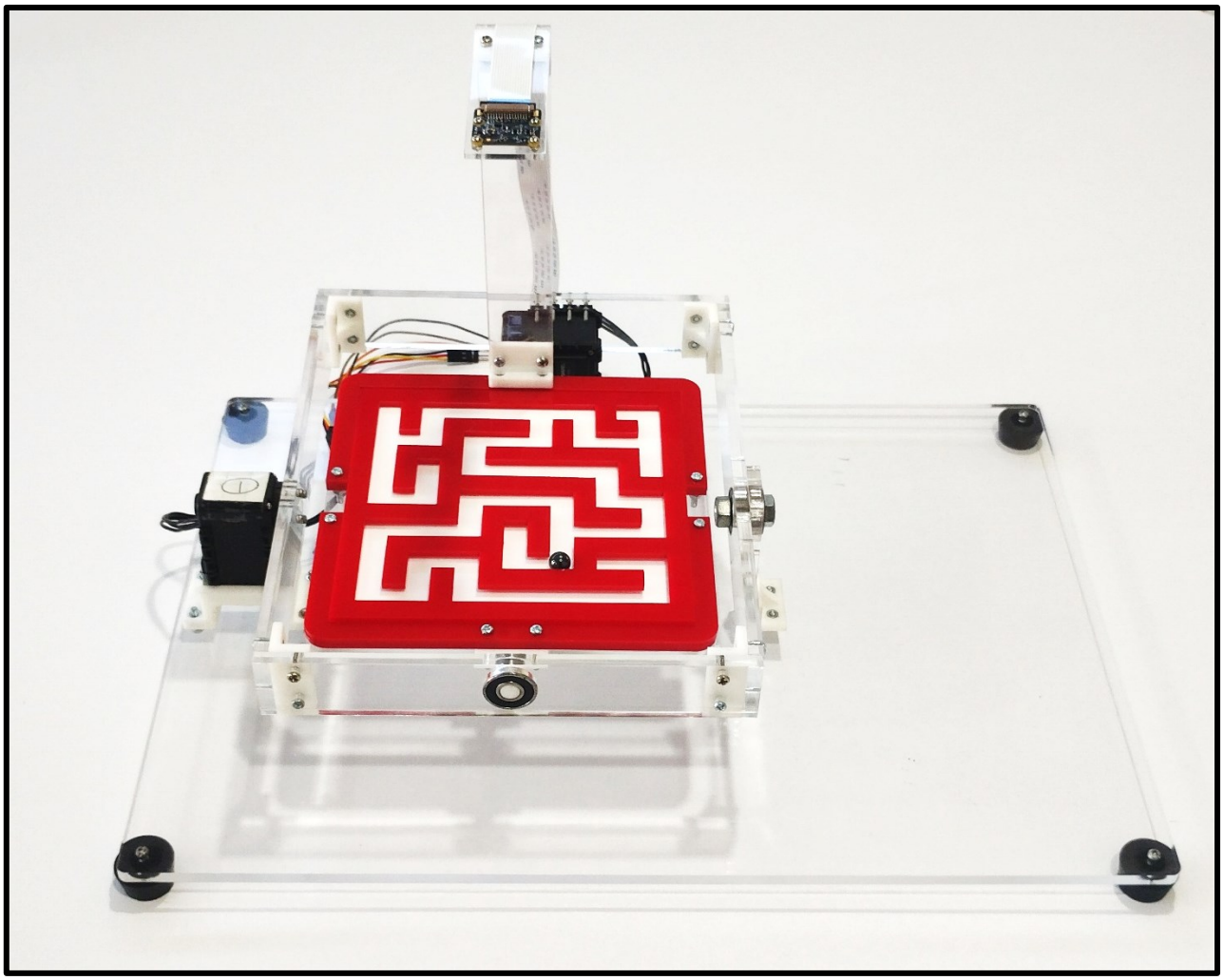


Figure 28: Final Product.

The final product is an autonomous maze-solving robot system designed with meticulous planning and research. Key components include interchangeable maze puzzles, a tilting platform with precise servo control, and simple electrical circuitry powered by a 12VDC, 5A power supply from a wall adapter.

The system integrates Raspberry Pi and PSOC microcontrollers for high-level decision-making and servo control, respectively. Communication protocols like SPI and UART facilitate efficient data exchange between components.

Computer vision capabilities, driven by the Raspberry Pi Camera Module V2, accurately detect the ball's position and maze layout. Path planning algorithms, such as RRT*, generate optimal paths while PID control ensures precise servo positioning.

The web-based user interface, developed using Flask and HTML/CSS/JavaScript, offers intuitive interaction and real-time monitoring. Users can initiate maze solving, monitor progress, and cancel operations as needed.

Overall, the system represents the integration of computer vision, path planning, and control systems to create an autonomous maze-solving robot.

7.2.3 Key features

Modularity and User Experience:

The system features Interchangeable Maze Designs. The system is designed to accept any maze configuration as long as the walls are red, and the dimensions are 15cm by 15cm by 3mm. The interchangeable design promotes creativity, allowing users to design and test their own maze configurations. This modular approach enhances user engagement and provides flexibility in testing different maze designs without hardware modifications.

Safety Mechanisms:

The system includes hard limits on the angles to prevent excessive tilting, ensuring the ball does not fall off the platform and preventing the platform from colliding into the base.

The system also automatically stops solving the maze if the ball is removed from the platform. It will resume solving if the ball is placed back on the platform at any position, ensuring seamless continuation of the maze-solving process.

Minimal Design:

The system uses a straightforward circuit design with minimal wiring, only three signal wires (SPI_CLK, SPI_MOSI, UART_DATA) in addition to power and ground connections. Hence, the design emphasizes software over hardware. This approach reduces costs by minimizing the number of physical components required. The simplicity of the circuit design makes the system easier to maintain and modify, supporting long-term usability and adaptability.

7.2.4 Testing and Results

To ensure the consistency and reliability of the autonomous ball in maze solving robot system, rigorous testing was conducted across various conditions and scenarios. The testing process included ball detection, maze layout detection, path planning and maze solving.

Ball Detection Testing:

Ball detection capabilities were evaluated under different lighting conditions and random platform orientations. The testing involved 20 runs each for scenarios with and without the ball, assessing the system's ability to accurately detect the ball. The results are shown in **Table 28**.

Table 28: Ball Detection Testing.

Lighting Condition	With ball			Without ball	
	Ball Detected Correctly (True Detection)	Ball Detected Incorrectly (False Detection)	No ball Detected (False Detection)	No ball Detected (True Detection)	Ball Detected (False Detection)
Controlled Lighting (Robotics Lab)	100%	0%	0%	100%	0%
Indoor Lighting (Sir Louis Matheson Library)	100%	0%	5%	100%	0%
Outdoor Lighting (Monash Clayton Campus Outdoor Bench with power outlet)	65%	25%	10%	70%	30%

The results indicate good ball detection performance under controlled and indoor lighting conditions. However, outdoor lighting poses challenges, with reduced accuracy due to environmental factors. It is important to note that outdoor lighting is limited to areas with wall outlets which is a limitation and will be addressed in **Section 7.4**.

Maze Layout Detection Testing:

The maze layout detection algorithm was tested to ensure accurate identification of maze configurations. Correct layout was assessed against predefined designs. False detection includes occluded paths or incorrect patches of wall or empty space. The testing involved 20 runs each and results are shown in **Table 29**.

Table 29: Maze Layout Detection Testing.

Lighting Condition	True Detection	False Detection
Controlled Lighting (Robotics Lab)	100%	0%
Indoor Lighting (Sir Louis Matheson Library)	100%	0%
Outdoor Lighting (Monash Clayton Campus Outdoor Bench with power outlet)	50%	50%

The system demonstrates high accuracy in maze layout detection under controlled and indoor lighting conditions. However, outdoor lighting impacts detection accuracy.

Moving forward, all tests will be conducted under indoor lighting conditions, the intended use-case scenario, to ensure consistent performance.

Path Planning Testing:

Path planning performance was evaluated by measuring average time taken to generate a path for different start and goal positions across three different maze designs: simple path, grid-based maze, and radial maze. The path planner (RRT*) was configured to use "Stopped when reached" flag with 20000 iterations. The testing involved 10 runs each and results are shown in **Table 30**.

Table 30: Path Planning Testing.

Maze Design	Average Time (s)
Simple Path	15
Grid-based	30
Radial	25

Figure 29 shows the system finding paths for the different maze designs.

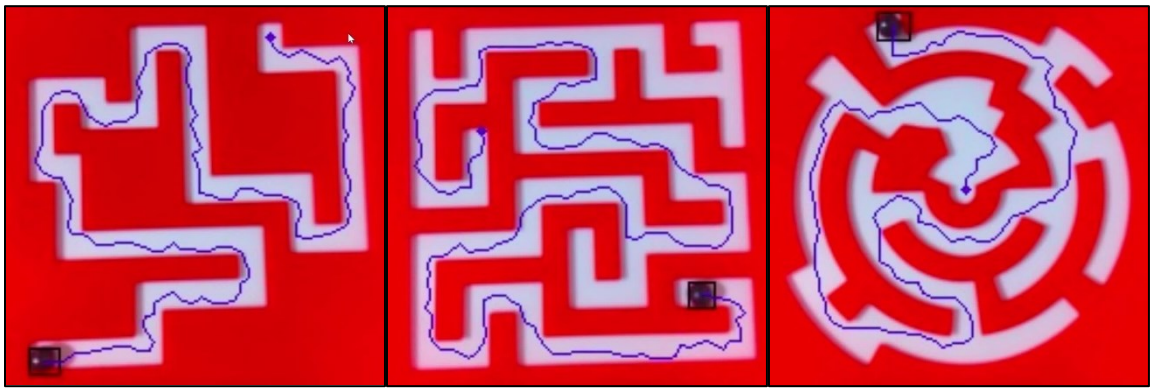


Figure 29: Different Maze Designs Path Planning.

The system achieves satisfactory path planning results, with average times ranging from 15 to 30 seconds across different maze designs. Nevertheless, it achieved 100% success rate in generating a feasible path.

Maze Solving Testing:

Each maze design was tested 10 times. The average time for maze solving and the number of runs requiring rerouting were recorded in **Table 31**.

Table 31: Maze Solving Testing.

Maze Design	Average Time (minutes)	Rerouting Required (out of 10 runs)
Simple Path	1	0
Grid-based	2.5	5
Radial	2	4

Longer solving times and rerouting instances suggest potential limitations in the controller design, impacting real-time responsiveness. Despite longer durations, the system achieved a 100% success rate in maze solving across all designs. Addressing controller design inefficiencies could improve overall performance and reduce solving time, especially for complex maze designs which is highlighted in **Section 7.4** .

7.3 Stakeholder feedback

Throughout the project, Mr. Michael Zenere provided invaluable guidance and feedback through weekly meetings every Monday. His insights were instrumental in multiple aspects of the project, leading to improvements. Some of the noteworthy feedback I have received throughout the project are highlighted in this section.

7.3.1 Servo Selection Feedback

Initially, I considered using a simple MG960 servo. Mr. Zenere suggested using the Dynamixel AX-12A servo due to his prior experience with it, noting its high speed and accuracy which is suitable for a tilting platform. The AX-12A was adopted, improving the system's performance significantly.

7.3.2 Platform Design Feeback

The original design was too bulky and heavy as it used aluminium. Mr. Zenere recommended using laser-cut acrylic for the platform and suggested using less bulky design choices where possible. This was suggested to save material and for more efficient and elegant design. **Figure 30** shows the progression of the platform design after multiple consultations with Mr. Zenere.

Figure 30: Iterative Designs of Robot.

7.3.3 Servo Controller Feedback

I initially planned to purchase the dedicated control board for the Dynamixel servos, which requires half-duplex communication that a Raspberry Pi cannot handle alone as shown in **Figure 31** [24]. This dedicated control boards, can be easily connected to the Raspberry Pi to control the servos and is well documented. However, the board cost AUD 40 which is costly and not customisable.

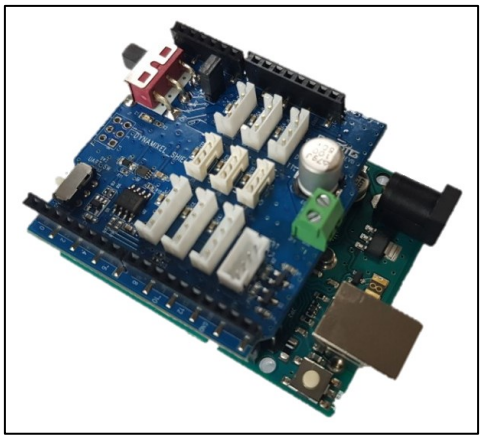


Figure 31: Dynamixel ax-12a Servo Shield.

Mr. Zenere advised using a PSOC to design the controller from scratch, which would be cheaper, provide a valuable learning experience and provide customisability. The PSOC-based controller was developed, exceeding the project requirements, and adding extra functionalities.

7.3.4 Maze Designs Feedback

I initially intended to limit the maze designs to grid-based patterns. Mr. Zenere suggested showcasing the robot's capabilities by not restricting the maze designs. The system now supports a variety of maze designs, highlighting its versatility.

7.3.5 PCB Design

I considered custom designing a PCB for the project. After consulting with Mr. Zenere, he recommended against this due to the simplicity of the wiring, which made manufacturing a PCB irresponsible in terms of sustainability. The project proceeded without a custom PCB, aligning with sustainable practices.

7.3.6 Feedback on the Final Product:

Positive Aspects:

- The project met all the requirements, achieving a 100% maze-solving rate.
- It stayed within the budget.
- The graphical user interface (GUI) is intuitive and accessible.
- Mr. Zenere was particularly impressed with the servo controller designed using PSOC.

Areas for Improvement:

- The main PID controller has limitations and could be improved.
- Solving time can be further optimized.

Overall, Mr. Zenere acknowledged the project's success, especially given the constraints of two semesters and a limited budget. His continuous feedback was essential in driving the project to its completion.

7.4 Future Improvements

Based on the feedback received and the results from rigorous testing, several future improvements have been identified. Two significant areas for enhancement are the controller design and portability of the system.

7.4.1 Controller Design

The current system exhibits relatively high solving times, particularly in complex scenarios. This is because the ball sometimes deviates from its intended path, especially in more complex maze designs. While the PID controller has been satisfactory, there is potential for improvement.

The system's software-based and modular coding approach allows for the addition of other control mechanisms without the need for removing or modifying the existing code. One potential improvement is the implementation of Model Predictive Control (MPC) [25]. MPC can handle control problems by predicting future behaviour and optimizing control actions accordingly as shown in **Figure 32**.

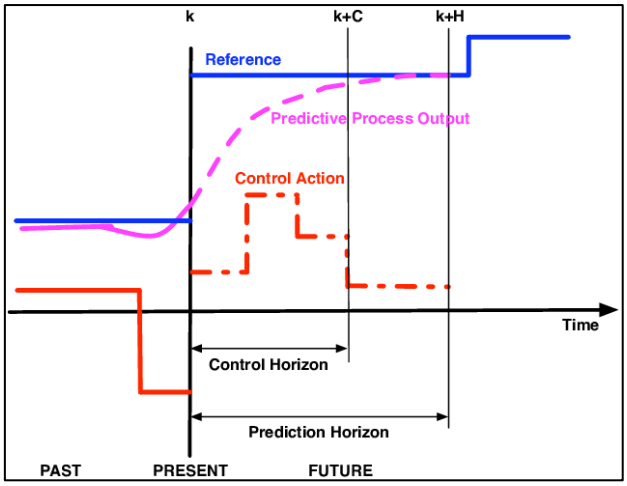


Figure 32: Model Predictive Control (MPC).

Unlike PID, which relies on current and past errors, MPC uses a model of the system to predict future states and compute control actions that optimize a performance criterion over a future time horizon.

7.4.2 Portability

The current system relies on a wall adapter, limiting its portability. Future versions could incorporate a battery pack to allow for cordless operation. Besides that, the current camera mount can be fragile and unstable during transportation. A foldable camera mount design, held in place with neodymium magnets, can reduce the height and provide greater stability during transport. **Figure 33** illustrates the proposed system that I designed in SolidWorks for the foldable camera mount.

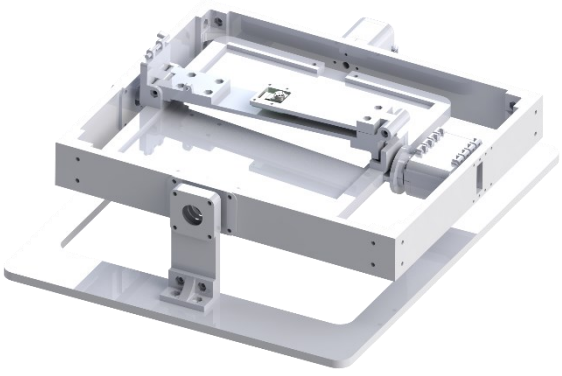


Figure 33: Foldable Camera Mount Design.

8 Conclusion

In conclusion, this project successfully developed an autonomous maze-solving robot system that integrates computer vision, path planning, and control mechanisms to navigate a ball through various maze designs. Through design and testing, the system demonstrated high reliability achieving a 100% success rate in maze-solving tasks.

The project successfully met all its objectives, as outlined in the **Table 32**.

Table 32: Objective Achievements.

No.	Objective	Achievement
1	Develop a vision-based control system using advanced image processing to detect the ball's position and maze configuration in real-time.	The system achieved over 95% accuracy in controlled and indoor lighting conditions, successfully detecting the ball and maze configuration in less than 2ms.
2	Implement sophisticated control algorithms to achieve precise movement and positioning of the ball within the maze.	The implemented PID control that ensured the ball consistently reached the goal within the specified tolerance of 5 mm
3	Ensure the robot can handle mazes with intricate designs and varying levels of complexity and adapt to dynamic environments.	The robot successfully solved mazes of varying complexity, including simple paths, grid-based mazes, and radial designs with 100% solve rate
4	Create a graphical user interface (GUI) for real-time visualization of the robot's progress and state.	The web-based user interface offers intuitive interaction and real-time monitoring.
5	Implement safety features to prevent accidents or damage during the robot's operation.	Comprehensive safety features were implemented and successfully tested, preventing excessive tilting and allowing for seamless continuation after ball removal
6	Focus on making the robot cost-effective without compromising functionality and performance.	The project maintained a cost under AUD 150, ensuring all functional and performance requirements were met

Other key accomplishments include:

- Modularity and User Engagement: The interchangeable maze design promotes user creativity and engagement, allowing for diverse testing scenarios without hardware modifications.
- Simplified Design: The minimalistic circuit design emphasizes software solutions, reducing costs and enhancing the system's adaptability for future improvements.

Despite the project's success, several areas for improvement were identified.

- The current PID controller exhibited limitations in complex scenarios leading to higher solving times and occasional path deviations. Implementing more advanced control algorithms, such as Model Predictive Control (MPC), could enhance the system's responsiveness and accuracy.
- Enhancing the system's portability with a battery pack and a foldable camera mount would increase its practicality for real-world applications.

Overall, this project successfully designed and prototyped a flexible and user-friendly autonomous ball in maze solving robot. It shows potential for further improvements, such as more sophisticated controllers and portable designs.

9 Reflection on Project Management

9.1 Project Scope

To ensure that the project meets the requirements and expectations, the project scope needs to be well defined. This will make sure that the project does not become too ambitious or lacklustre. In this section, a detailed list of in-scope and out-of-scope functionalities will be defined.

In the context of this “Ball in Maze Solving Robot” project, the scope encompasses the design, development, and testing of a robotic system capable of solving a ball-in-maze puzzle. The project will involve the creation of a robot that will autonomously manipulate the maze to guide a ball from the start position to the goal.

9.1.1 In-Scope Functionalities

1. Uses Camera sensor to detect maze layout, obstacles, and ball position.
2. Use Tracking to determine ball position.
3. Create a digital representation of the Maze.
4. Use an algorithm to determine optimal path for ball to reach target.
5. Coordinate the servo motors to control the tilting platform to manipulate the maze.
6. Display a user-friendly interface that shows relevant information.

9.1.2 Out-Of-Scope Functionalities

1. Implementing complex machine learning techniques for maze-solving is not considered.
2. The Maze digital representation will not be real time. A slight delay is to be expected.

9.2 Project Plan & Timeline

The Project timeline provides a comprehensive overview of the timeline of this Final year project. The duration of the project is from Jully 2023 to May 2024. The breaks are not included in the project timeline. **Figure 34** illustrates the original proposed timeline, while **Figure 35** shows the final timeline. Note that the dates are based on Monash semester dates.

Semester (Monash Calender)				2023											2024										
Week (Monash Calender)				1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12
No.	Task	Progess (%)	Duration (Weeks)																								
Research and Planning																											
1	Foundational Reading	100%	1																								
2	State of Field	90%	3																								
3	Project Planning and Scheduling	100%	1																								
4	Risk Assesement (SARAH)	90%	2																								
5	Sketch Posibble Designs	100%	3																								
6	CAD the design	90%	2																								
Prototyping and Experimenting																											
7	Experiment with Raspberry Pi and Electronics	0%	2																								
8	Experiment with different Maze Solving Algorithms	0%	3																								
9	Experiment with Computer Vission techniques	0%	3																								
10	Procure Materials	0%	1																								
11	Contrstruct Titlting Platform Prototype using simple materials	0%	2																								
12	Attach and intergrate electronics to the Prototype	0%	1																								
13	Test The tilting Platform range of motion	0%	1																								
14	Test the Camera and peform Calibration	0%	1																								
15	Test The touch screen display	0%	1																								
Mechanical Design																											
16	Draw a detailed sketches of the design with dimensions	0%	1																								
17	Designed all Parts in SOLIDWORKS	0%	2																								
18	Obtain Engineering and Assembly drawings	0%	1																								
19	Manufacture all custom parts via 3d printing or machining	0%	3																								
20	Assemble the Tilting Maze Platform	0%	3																								
Electrical Design																											
21	Design the Circuit required using software (LTSpice)	0%	2																								
22	Prototype Circuit on a breadboard	0%	2																								
23	Test and ensure power rating of components match simulation results	0%	2																								
24	Attach all electronic components to the robot	0%	2																								
25	Test all functionality of sensors, actuators and display	0%	2																								
26	Permenantly Solder the circuit and test it	0%	1																								
Software Design																											
27	Peform preprocesing on camera input	0%	2																								
28	detect maze layout, obstacles, and ball position	0%	2																								
29	Tracking to determine ball trajectory	0%	1																								
30	Create a digital representation of the Maze	0%	2																								
31	Use selected alorithm to determine optimal path for the Maze	0%	2																								
32	Implement a control system to control servos	0%	1																								
33	Display a user-friendly interface that shows relevant information	0%	3																								
Integration and testing																											
34	Peform Final testing for various cases	0%	1																								
35	Demontsrate the robot solving the maze	0%	2																								
36	Implement additional features based on testing	0%	2																								
37	Validate User interface	0%	2																								
38	Validate Automated Maze Solving Robot	0%	2																								
Submissions																											
39	Project Proposal	80%	5																								
40	Progress Report	0%	6																								
41	Project Poster	0%	6																								
42	Project Video	0%	6																								
43	Final Report	0%	12																								

Figure 34: Proposed Timeline.

Semester (Monash Calender)				Semester 2 2023											Semester 1 2024										
Week (Monash Calender)				1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12
No.	Task	Progess (%)	Duration (Weeks)																								
Research and Planning																											
1	Foundational Reading	100%	1																								
2	State of Field	100%	3																								
3	Project Planning and Scheduling	100%	1																								
4	Risk Assesement (SARAH)	100%	2																								
5	Sketch Posibble Designs	100%	3																								
6	CAD the design	100%	3																								
Prototyping and Experimenting																											
7	Experiment with Raspberry Pi and Electronics	100%	2																								
8	Experiment with different Maze Solving Algorithms	100%	3																								
9	Experiment with Computer Vission techniques	100%	3																								
10	Procure Materials	100%	1																								
11	Contrstruct Titlting Platform Prototype using simple materials	100%	2																								
12	Attach and intergrate electronics to the Prototype	100%	1																								
13	Test The tilting Platform range of motion	100%	1																								
14	Test the Camera and peform Calibration	100%	1																								
15	Test The touch screen display	100%	1																								
Mechanical Design																											
16	Draw a detailed sketches of the design with dimensions	100%	1																								
17	Designed all Parts in SOLIDWORKS	100%	1																								
18	Obtain Engineering and Assembly drawings	100%	1																								
19	Manufacture all custom parts via 3d printing or machining	100%	3																								
20	Assemble the Tilting Maze Platform	100%	4																								
Electrical Design																											
21	Design the Circuit required using software (LTSpice)	100%	2																								
22	Prototype Circuit on a breadboard	100%	1																								
23	Test and ensure power rating of components match simulation results	100%	1																								
24	Attach all electronic components to the robot	100%	1																								
25	Test all functionality of sensors, actuators and display	100%	1																								
26	Permenantly Solder the circuit and test it	100%	1																								
Software Design																											
27	Peform preprocesing on camera input	100%	4																								
28	detect maze layout, obstacles, and ball position	100%	4																								
29	Tracking to determine ball trajectory	100%	4																								
30	Create a digital representation of the Maze	100%	3																								
31	Use selected alorithm to determine optimal path for the Maze	100%	5																								
32	Implement a control system to control servos	100%	4																								
33	Display a user-friendly interface that shows relevant information	100%	3																								
Integration and testing																											
34	Peform Final testing for various cases	100%	3																								
35	Demontsrate the robot solving the maze	100%	3																								
36	Implement additional features based on testing	100%	2																								
37	Validate User interface	100%	1																								
38	Validate Automated Maze Solving Robot	100%	2																								
Submissions																											
39	Project Proposal	100%	5																								
40	Progress Report	100%	6																								
41	Project Poster	100%	6																								
42	Project Video	100%	6																								
43	Final Report	100%	9																								

Figure 35: Final Timeline.

9.3 Reflection on Project (500 Words)

9.3.1 Individual Performance

Overall, the project was a success, achieving its primary objectives and providing insights to areas of improvement. Throughout the project, I gained valuable learning experience in technical skills, project Management and self-development. The most prominent values that I have learned are highlighted as follows.

- **Time Management:** Effective time management was essential. Planning and adhering to the schedule ensured steady progress and timely completion.
- **Regular Meetings with Supervisor/Client:** Weekly meetings with my supervisor were invaluable for guidance and keeping the project on track. Feedback from these sessions was crucial for making informed decisions and identify potential issues early.
- **Importance of Team Members:** Although this was an individual project, the challenges I faced led me to appreciate team-based projects. I faced difficulties in debugging and hardware setup without team members. These challenges highlighted the benefits of teamwork, such as gaining additional perspectives, sharing the workload, and receiving support.

In summary, the project highlighted to me the importance of time management, regular supervision, and the challenges of working alone.

9.3.2 Improvement Strategies for future projects

For future projects, several improvement strategies can be implemented:

- **Early Identification of Potential Issues**: More rigorous initial testing of components, particularly communication protocols, could pre-empt some of the delays experienced.
- **Iterative Prototyping:** Adopting a more iterative prototyping approach can help in identifying and resolving design flaws earlier in the process.

9.3.3 Timeline Adjustments

Key Changes and Solutions:

- Prototyping Delays:
    - Reason: Communication with the servo was more complicated than anticipated.
    - Solution: Consulted and debugged with a Michael Zenere.
- Mechanical Design Delays:
    - Reason: Iterative design process with back-and-forth testing.
    - Solution: Continuous testing and refinement improved the final design.
- Electrical Circuit Early Completion:
    - Reason: Finished earlier due to minimal design considerations and effective prototyping.
    - Solution: Early and adequate testing of electronic components.
- Extended Software Design Phase:
    - Reason: extensive PID tuning, and technical complications in programming the path planner and controller. Integration issue where each section works separately but not together.

    - Solution: Focused efforts on software development and consulted Michael Zenere if facing a complication.

- Documentation Submissions:
    - Reason: Started slightly later than planned to prioritize physical build.
    - Outcome: Did not negatively affect report progress.

In summary, I've realized the importance of the research phase and the iterative nature of product design. Moving forward, I'll prioritize research and allocate more time for potential complications. I also acknowledge that hitting milestones as hard deadlines is unrealistic. Instead, I'll aim for a rough picture with buffer time for unexpected challenges.

9.3.4 Scope Adjustments

We added more complexity to the maze configurations tested such as a radial design, pushing the boundaries of our initial scope to enhance the robot's adaptability.

No major elements were removed from the original scope.

9.3.5 Task and Sub-Task Completion

Despite the timeline adjustments, all tasks and sub-tasks were completed. The planning phase ensured that we met all objectives, even if some milestones were reached later than initially planned.

10 References

I acknowledge the use of ChatGPT (https://chat.openai.com/) to refine the academic language and accuracy of my own work. On 26th May 2024 I submitted the sections of my report with the prompt “Improve the academic tone and accuracy of language, including grammatical structures, punctuation and vocabulary”. The output was then modified further to better represent my own tone and style of writing. No ideas or new content generated by AI technologies has been used in this assessment.

[1] L. Keselman, J. Born, N. Heravi, J. Moss, and M. Coad, "A Maze Bot," Dept. of Computer Science, Stanford Univ., and Dept. of Mechanical Engineering, Stanford Univ., Jun. 12, 2017.
[2] Admin, “The History of Dexterity & Maze Puzzles,” SiamMandalay, May 19, 2021. https://www.siammandalay.com/2021/05/19/the-history-of-dexterity-maze-puzzles/
[3] Puzzlemuseum.com, 2023. https://www.puzzlemuseum.com/faqs/oldestpz.htm
[4] J. M. Tsarapkina, M. M. Petrova, A. G. Mironov, I. M. Morozova, and O. B. Shustova, "Robotics as a basis for informatization of education in a children’s health camp," Amazonia Investiga, vol. 8, no. 20, pp. 115-123, Jun. 2019. [Online]. Available: https://www.amazoniainvestiga.info/index.php/amazonia/article/view/70
[5] Claudio, Giovanni & Agravante, Don & Spindler, Fabien & Chaumette, François. (2016). Dual arm manipulation and whole-body control with the humanoid robot Romeo by visual servoing.
[6] Erwanto, Rizky and Dwi Endah Kurniasih. “The effectiveness of puzzle therapy on cognitive functions among elderly with dementia at Balai Pelayanan Sosial Tresna Werdha (BPSTW) Yogyakarta, Indonesia.” Bali Medical Journal 9 (2020): 86.
[7] Poerio GL, Blakey E, Hostler TJ, Veltri T (2018) More than a feeling: Autonomous sensory meridian response (ASMR) is characterized by reliable changes in affect and physiology. PLoS ONE 13(6): e0196645. https://doi.org/10.1371/journal.pone.0196645
[8] L. Spacek, V. Bobál, and J. Vojtesek, "Maze Navigation on Ball & Plate Model," in Computer Science On-line Conference, 2017. [Online]. doi:10.1007/978-3-319-57264-2_21
[9] B.M. Arthayaet al., “Design and kinematic analysis of a two-DOF moving platform as a base for a car simulator,”Journal of Mechatronics, Electrical Power, and Vehicular Technology, vol. 13, no. 1, pp.48-59, July 2022, doi: https://doi.org/10.14203/j.mev.2022.v13.48-59
[10] D. Romeres, D. Jha, A. Dalla Libera, W. Yerazunis, and D. Nikovski, "Learning Hybrid Models to Control a Ball in a Circular Maze," Sep. 2018.
[11] M. O. A. Aqel, A. Issa, M. Khdair, M. ElHabbash, M. AbuBaker and M. Massoud, "Intelligent Maze Solving Robot Based on Image Processing and Graph Theory Algorithms," 2017 International Conference on Promising Electronic Technologies (ICPET), Deir El-Balah, Palestine, 2017, pp. 48-53, doi: 10.1109/ICPET.2017.15.
[12] A. Petruska, A. Mahoney, and J. Abbott, "Remote Manipulation With a Stationary Computer-Controlled Magnetic Dipole Source," IEEE Transactions on Robotics, vol. 30, pp. 1222-1227, Oct. 2014. doi: 10.1109/TRO.2014.2340111
[13] Jaime Gallardo-Alvarado, José Gallardo-Razo, Chapter 2 - Overview of mechanisms and robot manipulators, Editor(s): Jaime Gallardo-Alvarado, José Gallardo-Razo, In Emerging Methodologies and Applications in Modelling, Identification and Control, Mechanisms, Academic Press, 2022, Pages 17-32, ISBN 9780323953481, https://doi.org/10.1016/B978-0-32-395348-1.00011-9.
[14] Y. Patel and P. George, "Parallel Manipulators Applications—A Survey," Modern Mechanical Engineering, Vol. 2 No. 3, 2012, pp. 57-64. doi: 10.4236/mme.2012.23008.
[15] K. Zarzycki and M. Ławryńczuk, “Fast Real-Time Model Predictive Control for a Ball-on-Plate Process,” Sensors, vol. 21, no. 12, p. 3959, Jun. 2021, doi: 10.3390/s21123959.
[16] Bhaskar Dasgupta, T.S. Mruthyunjaya, The Stewart platform manipulator: a review, Mechanism and Machine Theory, Volume 35, Issue 1, 2000, Pages 15-40, ISSN 0094-114X, https://doi.org/10.1016/S0094-114X(99)00006-3
[17] Liu, XJ., Wang, J., Oh, KK. et al. A New Approach to the Design of a DELTA Robot with a Desired Workspace. Journal of Intelligent and Robotic Systems 39, 209–225 (2004). https://doi.org/10.1023/B:JINT.0000015403.67717.68

[18] K. -j. Seong, H. -g. Kang, B. -y. Yeo and H. -p. Lee, "The Stabilization Loop Design for a Two-Axis Gimbal System Using LQG/LTR Controller," 2006 SICE-ICASE International Joint Conference, Busan, Korea (South), 2006, pp. 755-759, doi: 10.1109/SICE.2006.315268.
[19] Nadour, M., Cherroun, L. Using flood-fill algorithms for an autonomous mobile robot maze navigation. Int J Syst Assur Eng Manag 13, 546–555 (2022). https://doi.org/10.1007/s13198-022-01630-4
[20] S. Alamri, H. Alamri, W. Alshehri, S. Alshehri, A. Alaklabi, and T. Alhmiedat, "An Autonomous Maze-Solving Robotic System Based on an Enhanced Wall-Follower Approach," Machines, vol. 11, no. 2, p. 249, 2023. [Online].doi: https://doi.org/10.3390/machines11020249
[21] Noreen, Iram & Khan, Amna & Habib, Zulfiqar. (2016). Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions. International Journal of Advanced Computer Science and Applications. 7.10.14569/IJACSA.2016.071114.
[22] A. Orthey, C. Chamzas, and L. E. Kavraki, “Sampling-Based Motion Planning: A Comparative Review,” arXiv.org, Sep.22, 2023. https://arxiv.org/abs/2309.13119.
[23] Flask, “Welcome to Flask — Flask Documentation (3.0.x),” flask.palletsprojects.com, 2010. https://flask.palletsprojects.com/en/3.0.x/
[24] Robotis “Dynamixel Shield” ROBOTIS e-Manual. https://emanual.robotis.com/docs/en/parts/interface/dynamixel_shield
[25] Garcia-Nieto Rodriguez, Sergio & Salcedo, José & Martínez, Miguel & Reynoso-Meza, Gilberto. (2010). FUZZ-IEEE Iterative Discrete Forward-Backward Fuzzy Predictive Control. 1 - 7. 10.1109/FUZZY.2010.5584319.
[26] Monash University, "Occupational Health, Safety & Wellbeing Policy Version: 3.3," May 2021.
[27] United Nations, “Goal 4 | Ensure Inclusive and Equitable Quality Education and Promote Lifelong Learning Opportunities for All,” United Nations, 2022. https://sdgs.un.org/goals/goal4
[28] “Recycling Aluminium | The Australian Aluminium Council,” Australian Aluminium Council Ltd, Oct. 19, 2017. https://aluminium.org.au/how-aluminium-is-made/recycling-aluminium-chart/
[29] Dierk Raabe, Dirk Ponge, Peter J. Uggowitzer, Moritz Roscher, Mario Paolantonio, Chuanlai Liu, Helmut Antrekowitsch, Ernst Kozeschnik, David Seidmann, Baptiste Gault, Frédéric De Geuser, Alexis Deschamps, Christopher Hutchinson, Chunhui Liu, Zhiming Li, Philip Prangnell, Joseph Robson, Pratheek Shanthraj, Samad Vakili, Chad Sinclair, Laure Bourgeois, Stefan Pogatscher, Making sustainable aluminum by recycling scrap: The science of “dirty” alloys, Progress in Materials Science,Volume 128, 2022, 100947, ISSN 0079-6425,https://doi.org/10.1016/j.pmatsci.2022.100947.(https://www.sciencedirect.com/science/article/pii/S0079642522000287)
[30] “Precious Plastic Monash,” Precious Plastic Monash. https://www.preciousplasticmonash.com/.
[31] “Precious Plastic Melbourne,” Precious Plastic Melbourne. https://www.plastic.org.au/

11 Appendices

11.1 Appendix A: Project Risk Assessment

51333	RISK DESCRIPTION	STATUS	TREND	CURRENT	RESIDUAL
	ENG4701_CL_2023_S2_AvvienashJaganathan_MazeSolvingRobot	Draft		Medium	Medium

RISK TYPE
1. Activity or Task Based Risk Assessment

RISK OWNER	RISK IDENTIFIED ON	LAST REVIEWED ON	NEXT SCHEDULED REVIEW
AVVIENASH JAGANATHAN	06/08/2023		

RISK FACTOR(S)	EXISTING CONTROL(S)	CURRENT	PROPOSED CONTROL(S)	TREATMENT OWNER	DUE DATE	RESIDUAL
As the project involves a rolling ball within a maze, there is a risk of the ball escaping the maze and causing tripping or injury to surrounding objects or people.	Control: Promptly pick up fallen components and clear the floor as soon as possible Control Effectiveness:	Low	No Control:			Low
Incorrect wiring or improper use of power sources may lead to electric shocks or fires.	Control: Wires powering electrical equipment in the lab are fully insulated. Control Effectiveness:	Medium	Always design and run a virtual simulation of a circuit before attempting to build the physical circuit.	AVVIENASH JAGANATHAN	06/08/2023	Low
Fingers or hands may get caught in between the moving parts of the robot such as the servo motor causing injuries	Control: Ensure no body parts are caught during the operation of the robot. Control Effectiveness:	Medium	When working on the Robot ensure the robot is switched off.	AVVIENASH JAGANATHAN	25/08/2023	Low
			Always ensure there is a working emergency stop button	AVVIENASH JAGANATHAN	25/08/2023	
The maze designed may have sharp corner or Protrusions which is a hazard that may cause injuries	Control: Be careful when handling the robot and watch out for sharp edges Control Effectiveness:	Low	Ensure that all designed parts are rounded and not sharp.	AVVIENASH JAGANATHAN	07/08/2023	Low
Chocking Hazards, from the small steel ball in the maze	Control: Promptly pick up fallen components and clear the floor as soon as possible Control Effectiveness:	Low	Build a cover to ensure that the moving parts (ball) don't escape during the operation	AVVIENASH JAGANATHAN	07/08/2023	Low
Burn Hazard - The use of soldering irons and hot glue guns may pose a risk of burns to project participants.	Control: Place the soldering iron or any other hazardous equipment back onto its designated stand when not in use Control Effectiveness:	Medium	No Control:			Medium

	Control: Ensure there are no flammable materials while handling these hazardous materials Control Effectiveness: _____ Control: Take note of the nearest Fire extinguisher and fire escape route. Control Effectiveness:					
Damage to ergonomics and Eye Strain due to prolonged use of Computer to program the robot	Control: There are no current control measures Control Effectiveness:	Medium	Take frequent breaks when using the computer for long periods of time	AVVIENASH JAGANATHAN	07/08/2023	Low
Using power tools and machining tools have sharp edges and may cause injuries	Control: Ensure to use the appropriate personal protective equipment for the tool being used. Control Effectiveness: _____ Control: If unfamiliar or unqualified to use the machine, ask the professional for help. Control Effectiveness:	Medium	Where possible, find alternative methods to build complex parts such as 3d printing.	AVVIENASH JAGANATHAN	07/08/2023	Medium
Health hazard of contracting deseases such as COVID -19 when working on campus	Control: Stay at home if feeling unwell. Control Effectiveness: _____ Control: Always ensure clean hands before handling the robot. Control Effectiveness:	Low	No Control:			Low

11.2 Appendix B: Risk Management Plan

To ensure a successful completion and development of this final year project, a comprehensive evaluation of the potential risk that could impact the project's progress, safety, and overall success. Therefore, by identifying and assessing these risks and hazards, mitigation strategies could be implemented to reduce the likelihood and impact of adverse events.

This section outlines both the Occupational Health and Safety (OHS) risks, which pertain to the well-being of the individuals involved in the project, as well as Non-Occupational Health and Safety (Non-OHS) risks, which encompass several factors that could influence project outcomes that does not affect the health and safety of the team.

Each risk is analysed in terms of its likelihood, consequences, initial risk level, mitigation strategies, and the residual risk level after mitigation. This structured approach to risk assessment ensures that the project is well-prepared to address potential challenges and maintain a safe and efficient work environment throughout its various phases.

The risk management plan considers the hazards and risks associated with all the project activities. The definition for hazards and risk are:

Hazard: A source of potential harm or a situation with potential to cause harm or loss.

Risk: Probability of occurrence of an event that could cause a specific level of harm or loss to people, property, environment and financially over a time frame.

Table 33 is the risk likelihood scale and its definition. The description has explained the impact level of each category.

Table 33: Risk Likelihood Categories.

Likelihood Categories	Definitions
Highly Likely	will occur in most circumstances when the activity is undertaken (greater than 90% chance of occurring)
Likely	will probably occur in most circumstances when the activity is undertaken (51 to 90% chance of occurring)
Possible	might occur when the activity is undertaken (21 to 50% chance of occurring)
Unlikely	could happen at some time when the activity is undertaken (1 to 20% chance of occurring)
Rare	may happen only in exceptional circumstances when the activity is undertaken (less than 1% chance of occurring)

Table 34 is the risk consequence scale and its definition. The description of each category is based on Monash University OHS Risk Assessment guideline (Monash University, 2021) [26] .

Table 34: Risk Consequence Categories

Consequence Categories	Definitions
Severe	Completion of projects becomes impossible.
Major	Not able to proceed to another task / next phase. Completion of the project is delayed.
Moderate	Completion of tasks may be delayed by several weeks. May affect the completion of another task.
Minor	Completion of tasks may be delayed by several days. Will not affect the completion of another task.
Insignificant	Minor inconvenience. It will not affect the project timeline.

Table 35 shows the risk level of a project hazard in terms of risk likelihood and risk consequence. Risk level classifications are based on Monash University OHS Risk Assessment Guideline (Monash University, 2021).

Table 35: Risk Assessment Matrix

Consequence Likelihood	Insignificant	Minor	Moderate	Major	Severe
Rare	N	L	L	M	M
Unlikely	L	L	M	M	H
Possible	L	M	M	H	H
Likely	M	M	H	H	E
Highly Likely	M	H	H	E	E

N	Negligible
L	Low Risk
M	Medium Risk
H	High Risk
E	Extreme Risk

A full detailed report on the non-OHS risks of this project has been described in **Table 36**.

Table 36: Non-OHS Risk Assessment

Risk Assessment					Risk Control			
Hazard	**Risk Description**	**Likelihood**	**Consequence**	**Risk**	**Mitigation**	**Likelihood**	**Consequence**	**Residual Risk**
Delayed delivery of electrical components	Leads to project delays and hinder prototype development.	Likely	Major	H	Maintain communication with component supplier. Obtain part form local companies to ensure quick delivery	Possible	Moderate	M
Incorrect Specifications of ordered components	Parts may not function as required. Servo Motor may not have enough torque. Component may not be compatible	Possible	Major	H	Verify specification before placing an ordering. Double check specifications on receipt	Unlikely	Moderate	M
Faulty Actuators such as servo motors	Could lead to improper functioning of the robot	Possible	Major	H	Order from reputable source. test component upon arrival	Unlikely	Moderate	M
Faulty camera sensors	May result in inaccurate visual data collection	Possible	Major	H	Order from reputable source. test component upon arrival	Unlikely	Moderate	M
Insufficient current draw from power supply	Actuators might not perform to its maximum capacity	Unlikely	Major	M	Ensure Power supply meets requirements and monitor voltage and current draw	Rare	Moderate	L
Incorrect Dimensions of 3d printed or machined components	Parts may not fit together, leading to assembly issues and project delays.	Likely	Moderate	H	Double check designs and dimensions before manufacturing	Possible	Moderate	M
Data lost or Corruption of software or Code	Could lead to loss of work and delays in development. May need to start over from scratch	Possible	Moderate	M	Regularly back up code and push to GitHub. Use version control. Ensure computer is virus free	Unlikely	Minor	L

11.3 Appendix C: Sustainability Plan

Sustainability in engineering is a crucial consideration, and this project to build an Autonomous Ball-In-Maze Solving Robot aligns particularly well with

United Nations Sustainable Development Goal (UN SDG) 4: Quality Education

UN SDG 4 aims to "ensure inclusive and equitable quality education and promote lifelong learning opportunities for all." (**see Figure 36**) [27]. The associated targets and key indicators emphasize the importance of access to quality education, enhancing relevant skills, and ensuring an inclusive learning environment.

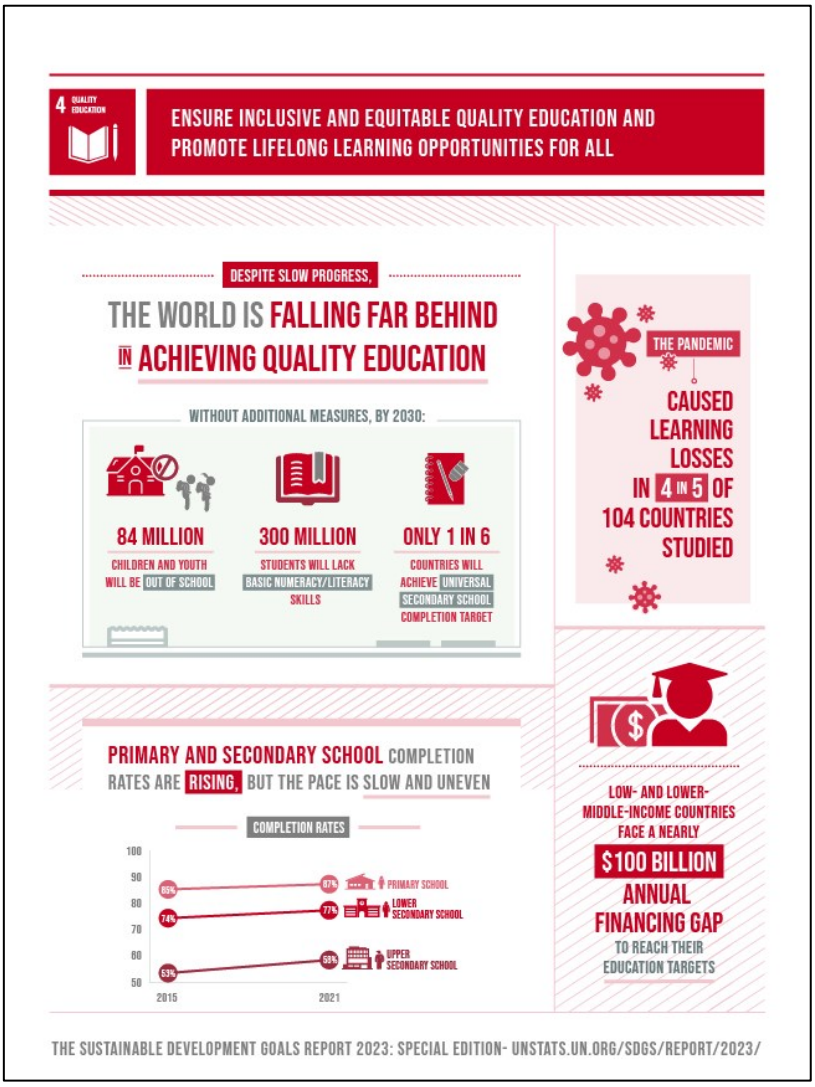


Figure 36: United Nations Sustainable Development Goal (UN SDG) 4: Quality Education.

The current scope of the project is not at a commercial level yet. However, in this section, the assumption is made that eventually this project's research and product will be used at an international and commercial scale.

Sustainable engineering, in the context of this project, contributes to achieving these goals through the following considerations:

Target 4.1: Primary and Secondary Education

One of the key targets of UN SDG 4 is to ensure that all children, regardless of gender, have access to free, equitable, and quality primary and secondary education. Our project contributes to this target by highlighting the capabilities of robotics systems for educational purposes. By providing a robot that can solve the Ball-In-Maze puzzle, we create an engaging and interactive educational tool. It encourages students, particularly in primary and secondary education, to explore the principles of robotics, computer vision, and problem-solving.

Target 4.4: Relevant Skills for Employment

Ensuring that youth and adults have relevant skills for employment, decent jobs, and entrepreneurship is another key target. Our project promotes skills development in robotics, computer vision, and control algorithms. These skills are highly relevant in today's job market, particularly in fields related to automation and technology. Students and learners who engage with our robot gain exposure to these practical, job-ready skills.

Target 4.7: Promoting Sustainable Development

One of the vital aspects of UN SDG 4 is to ensure that all learners acquire the knowledge and skills needed to promote sustainable development. Our project aligns with this target by sourcing from sustainable materials and developed sustainably.

The main materials used in the construction of the project is Aluminium (sheets and extrusions) and PLA (3d printed parts).

Aluminium can be recycled repeatedly without compromising its structural integrity or properties. This characteristic makes it a highly sustainable material choice. Notably, recycling aluminium consumes up to 95% less energy compared to producing new aluminium from raw materials. It is an energy-efficient and environmentally friendly choice. In fact, approximately 75 percent of all the aluminium ever produced is still in use today as it can be recycled endlessly without compromising any of its unique properties or quality [28] [29]. (**see Figure 37**)

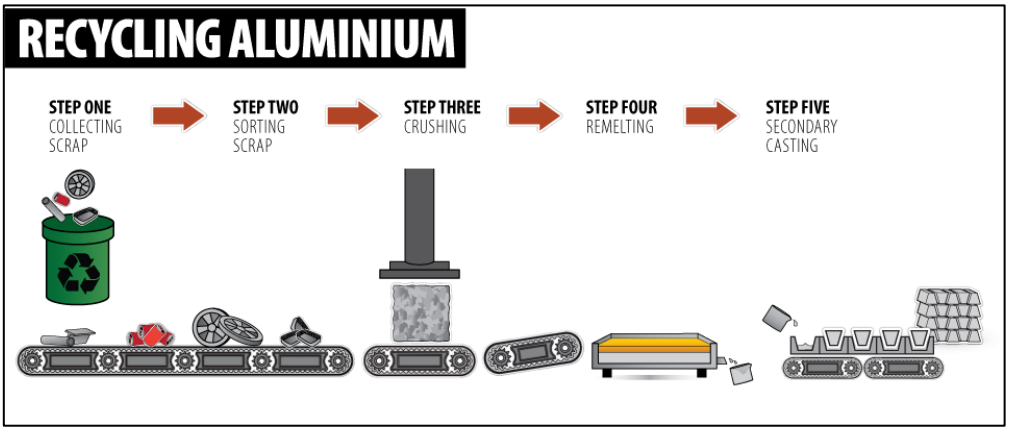


Figure 37: Process of Recycling Aluminium

PLA, chosen for its cost-effectiveness and lightweight qualities, serves as a key component in our project. However, we ensure responsible disposal and recycling. Monash University is actively involved in recycling all used PLA. They provide discarded PLA to an Engineering Student team known as Precious Plastic Monash. This team repurposes waste PLA into various products, including chairs and accessories. This collaborative effort ensures that any prototypes, mistakes, or unused PLA do not contribute to environmental waste [30] [31] .

Target 4.a: Inclusive and Effective Learning Environments

The project promotes inclusive and effective learning environments by introducing robotics and technology in education. By designing a safe and engaging robot that can solve puzzles, we ensure that the learning environment is inclusive and safe. The project is designed to be safe for children to use and a detailed Risk Assessment has been performed.

Some key steps to ensure that the final product is safe include:

1. Ensuring that there are no sharp corners.
2. Limiting the speed of the actuators to prevent sudden movements.
3. Designing a transparent cover to ensure that the maze and ball are visible yet to a potential hazard.

Target 4.c: Qualified Teachers and Teacher Training

While our project does not directly address teacher training, it can serve as a resource for educators. Teachers can use the Autonomous Ball-In-Maze Solving Robot as a practical educational tool to demonstrate engineering and robotics concepts in the classroom.

In conclusion, our project encourages problem-solving, critical thinking, and creativity, which are essential skills for lifelong learning and success in a rapidly changing world. By showcasing the potential of robotics in education, we contribute to building a foundation for quality education and sustainable development.

Another important aspect of the sustainability of the project is the social, cultural, health, safety, environmental and economic considerations which is neatly summarized in the **Table 37**.

Table 37: Sustainability Considerations.

Consideration	Key Points
Social	Reducing Stress: Design the robot to provide entertainment and stress relief for users.
	Skill Improvement: Encourage users to enhance their problem-solving and motor skills.
	Inspiration: Aim to inspire creativity and curiosity in users, especially among younger audiences.
	Cultural Sensitivity: Ensure that the robot's design and behaviour respect diverse cultural norms.
Health	Safety: Prioritize user safety by implementing features to prevent accidents and injuries.
	Ergonomics: Design the robot with user comfort in mind to prevent physical strain or discomfort.
Environmental	Carbon Emission Calculation: Monitor and minimize the robot's carbon footprint throughout its lifecycle.
	Material Reuse: Incorporate recyclable and reusable materials to reduce waste.
	Energy Efficiency: Optimize power consumption to minimize energy usage and environmental impact.
	Sustainable Materials: Choose eco-friendly materials that are responsibly sourced and manufactured.
Economic	Cost-Effectiveness: Strive to keep production and maintenance costs reasonable for affordability.
	Local Economic Support: If feasible, support local businesses and labour during the robot's production.
	Resource Conservation: Minimize resource consumption and waste to contribute to sustainability.

11.4 Appendix D: Generative AI Statement

Avvienash Jaganathan <ajag0007@student.monash.edu>

Generative AI use in FYP B (ENG4702)
1 message

Google Forms <forms-receipts-noreply@google.com>
To: ajag0007@student.monash.edu

26 May 2024 at 11:07

Thanks for filling in Generative AI use in FYP B (ENG4702)

Here's what was received.

Edit response

Generative AI use in FYP B (ENG4702)

The responses to this form will need to be copied and put into an appendix in your Final Report.

Email *

ajag0007@student.monash.edu

Name *

Avvienash Jaganathan

Campus

- (●) Clayton
- ( ) Malaysia

Host Department

- ( ) Chemical and Biological Engineering
- ( ) Civil Engineering
- ( ) Electrical and Computer Systems Engineeing
- ( ) Materials Science Engineering
- ( ) Mechanical and Aerospace Engineering
- ( ) Software Engineering
- (●) Robotics and Mechatronics Engineering

Supervisor

Michael Zenere

This project has been conducted using AI tools *

- ( ) In this assessment, there will be no use of generative artificial intelligence (AI). All content in relation to the assessment task has been produced by the authors.
- (●) In this assessment, the following generative AI will be used for the purposes nominated in part 2. (Please note: any use of generative AI must be appropriately acknowledged - see Learn HQ)
- ( ) In this assessment, AI writing assistants (e.g., Grammarly, Writesonic, Quillbot, Microsoft Editor) will be the only form of Generative AI used.
- ( ) This project involves the development or authoring of Unique Generative AI, Unique operation of commercially available Generative AI OR Unique non-generative AI (Machine Learning, Artificial Neural Network, Logistic Regression, etc.)

AI Tools used

In question 1, you answered you were using generative AI for the purposes nominated in part 2. Insert names of AI tools, or types of tools (e.g. image generators/text generators) you used.
(Please note: any use of generative AI must be appropriately acknowledged - see Learn HQ)

ChatGPT

How has the technology be used?

How did you use this technology? *

- [ ] As a fundamental aspect of the study (i.e., this is a study centred on generative AI, human interaction, associated ethics, etc.)
- [ ] Audio Transcription
- [ ] Coding/Scripting
- [ ] For the operation of robotics
- [ ] Generation of novel content - Datasets
- [ ] Generation of novel content - Graphics/Images
- [ ] Generation of novel content - Video
- [ ] Generation of novel content - Writing
- [ ] Idea generation
- [ ] Initial research
- [ ] Machine Language Translation
- [ ] Mathematics
- [x] Paraphrasing
- [x] Proofreading
- [ ] Text Analytics
- [ ] Text Summarisation
- [ ] Thematic analysis
- [ ] Visualisation (of data)
- [ ] Writing assistance
- [ ] Other:

How was the Generative AI response validated?

I submitted the sections of my report with the prompt “Improve the academic tone and accuracy of language, including grammatical structures, punctuation and vocabulary”. The output was then modified further to better represent my own tone and style of writing.

Permissions

The use of Generative AI has been discussed with and approved by my academic supervisor. *

- (•) Yes
- ( ) No

End

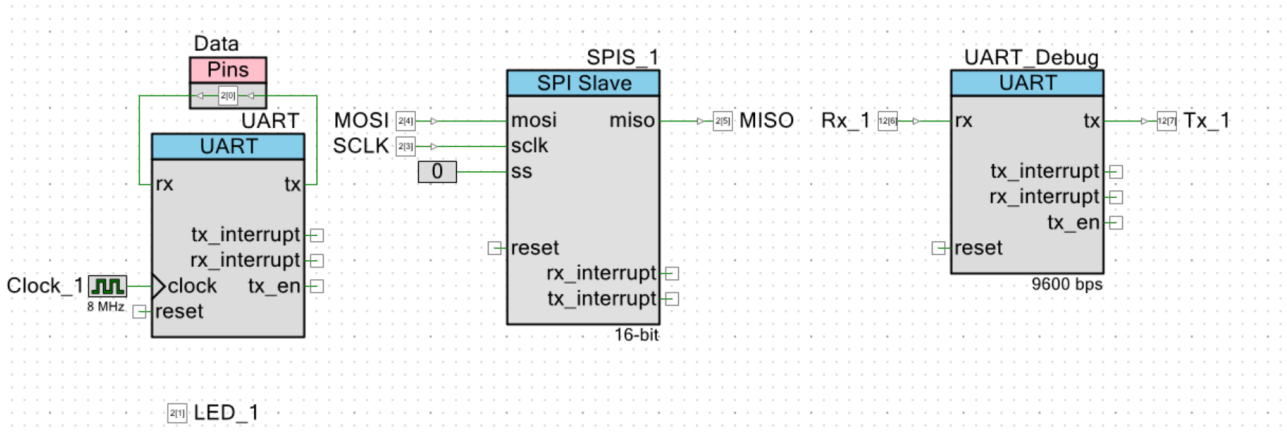
Thank you for completing this form - your responses will be emailed to you for your Progress Report

11.5 Appendix E: Raspberry Pi Code

The complete code for the Ball-In-Maze Solving robot is available in the public GitHub repository. You can access it at the following link: https://github.com/Avvienash/Maze-Solving-Robot

11.6 Appendix F: PSOC Code for Dynamixel Control

11.6.1 Top Design



11.6.2 Main.c

```
#include <project.h>

// EEPROM AREA  ///////////////////////////////////////////////////////////
#define AX_MODEL_NUMBER_L           0
#define AX_MODEL_NUMBER_H           1
#define AX_VERSION                  2
#define AX_ID                       3
#define AX_BAUD_RATE                4
#define AX_RETURN_DELAY_TIME        5
#define AX_CW_ANGLE_LIMIT_L         6
#define AX_CW_ANGLE_LIMIT_H         7
#define AX_CCW_ANGLE_LIMIT_L        8
#define AX_CCW_ANGLE_LIMIT_H        9
#define AX_SYSTEM_DATA2             10
#define AX_LIMIT_TEMPERATURE        11
#define AX_DOWN_LIMIT_VOLTAGE       12
#define AX_UP_LIMIT_VOLTAGE         13
#define AX_MAX_TORQUE_L             14
#define AX_MAX_TORQUE_H             15
#define AX_RETURN_LEVEL             16
#define AX_ALARM_LED                17
#define AX_ALARM_SHUTDOWN           18
#define AX_OPERATING_MODE           19
#define AX_DOWN_CALIBRATION_L       20
#define AX_DOWN_CALIBRATION_H       21
#define AX_UP_CALIBRATION_L         22
#define AX_UP_CALIBRATION_H         23

      // RAM AREA  ////////////////////////////////////////////////////////////
#define AX_TORQUE_ENABLE            24
#define AX_LED                      25
#define AX_CW_COMPLIANCE_MARGIN     26
#define AX_CCW_COMPLIANCE_MARGIN    27
#define AX_CW_COMPLIANCE_SLOPE      28
#define AX_CCW_COMPLIANCE_SLOPE     29
#define AX_GOAL_POSITION_L          30
#define AX_GOAL_POSITION_H          31
#define AX_GOAL_SPEED_L             32
#define AX_GOAL_SPEED_H             33
#define AX_TORQUE_LIMIT_L           34
#define AX_TORQUE_LIMIT_H           35
```

```
#define AX_PRESENT_POSITION_L       36
#define AX_PRESENT_POSITION_H       37
#define AX_PRESENT_SPEED_L          38
#define AX_PRESENT_SPEED_H          39
#define AX_PRESENT_LOAD_L           40
#define AX_PRESENT_LOAD_H           41
#define AX_PRESENT_VOLTAGE          42
#define AX_PRESENT_TEMPERATURE      43
#define AX_REGISTERED_INSTRUCTION   44
#define AX_PAUSE_TIME               45
#define AX_MOVING                   46
#define AX_LOCK                     47
#define AX_PUNCH_L                  48
#define AX_PUNCH_H                  49

    // Status Return Levels
//////////////////////////////////////////////////////////////
#define AX_RETURN_NONE              0
#define AX_RETURN_READ              1
#define AX_RETURN_ALL               2

    // Instruction Set ////////////////////////////////////////////////////////////
#define AX_PING                     1
#define AX_READ_DATA                2
#define AX_WRITE_DATA               3
#define AX_REG_WRITE                4
#define AX_ACTION                   5
#define AX_RESET                    6
#define AX_SYNC_WRITE               131

       // Specials ////////////////////////////////////////////////////////////
#define OFF                         0
#define ON                          1
#define LEFT                                0
#define RIGHT                       1
#define AX_BYTE_READ                1
#define AX_BYTE_READ_POS            2
#define AX_BYTE_READ_SPEED          2
#define AX_RESET_LENGTH                     2
#define AX_ACTION_LENGTH                2
#define AX_ID_LENGTH                4
#define AX_LR_LENGTH                4
#define AX_SRL_LENGTH               4
#define AX_RDT_LENGTH               4
#define AX_LEDALARM_LENGTH          4
#define AX_SALARM_LENGTH            4
#define AX_TL_LENGTH                4
#define AX_VL_LENGTH                6
#define AX_CM_LENGTH                6
#define AX_CS_LENGTH                6
#define AX_CCW_CW_LENGTH            8
#define AX_BD_LENGTH                4
#define AX_TEM_LENGTH               4
#define AX_MOVING_LENGTH            4
#define AX_RWS_LENGTH               4
#define AX_VOLT_LENGTH              4
#define AX_LED_LENGTH               4
#define AX_TORQUE_LENGTH            4
#define AX_POS_LENGTH               4
#define AX_GOAL_LENGTH              5
#define AX_MT_LENGTH                5
#define AX_PUNCH_LENGTH             5
#define AX_SPEED_LENGTH             5
#define AX_GOAL_SP_LENGTH           7
#define AX_ACTION_CHECKSUM              250
#define BROADCAST_ID                254
#define AX_START                    255
#define AX_CCW_AL_L                 255
#define AX_CCW_AL_H                 3
#define TIME_OUT                    10
```

```
#define TX_MODE                     1
#define RX_MODE                     0
#define LOCK                        1
#define ERROR_CHECK                 1


/// FUNCTIONS DECLARATIONS
// Debug
void debugPrint(const char *str);
void debugPrintInt(int n);

// Write
void LED_Control(uint8_t Status, uint8_t ID);
void Set_ID(uint8_t new_ID, uint8_t ID);
void Move(uint16_t Position, uint8_t ID);
void MoveSpeed(uint16_t Position, uint16_t Speed, uint8_t ID);
void TorqueStatus(uint8_t Status, uint8_t ID);
void SetReturnLevel(uint8_t level, uint8_t ID);
void reset(uint8_t ID);
void ping(uint8_t ID);
void MoveSpeedRW(uint16_t Position, uint16_t Speed, uint8_t ID);
void action(void);


//Read Error for Write Command
int ReadError(void);

// Read
int ReadVoltage(uint8_t ID);
int ReadTemperature(uint8_t ID);
int ReadPosition(uint8_t ID);
int ReadSpeed(uint8_t ID);



/// MAIN FUNCTION
int main(void)
 {
    CyGlobalIntEnable; /* Enable global interrupts. */

    // OFF LED
    LED_1_Write(0);

    // UART INITIALISATION
    UART_Start();
    UART_Debug_Start();
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    // SPI INITIALISATION
    SPIS_1_Start();
    char hexString[5];
    SPIS_1_ClearTxBuffer();
    SPIS_1_ClearRxBuffer();
    SPIS_1_ClearFIFO();


    // DYNA IDS DECLARATRION // ID 0XFE  Broadcasting ID
    uint8_t ID_0 = 0;
    uint8_t ID_1 = 1;

    // The Range for the Motors
    int flat_0 = 515;
    int flat_1 = 48;
    int max_input = 50;


    LED_1_Write(1);
    MoveSpeed(flat_0,100,ID_0);
    MoveSpeed(flat_1,100,ID_1);
```

```
LED_Control(1,1);
LED_Control(1,0);


// Declare variables for storing extracted information
uint16 receivedData;
uint8 sign_0;
uint8 sign_1;
uint8 value_0;
uint8 value_1;

int input_0;
int input_1;

for (;;)
{
    if(SPIS_1_ReadRxStatus() & SPIS_1_STS_RX_FIFO_NOT_EMPTY)
    {
        receivedData = SPIS_1_ReadRxData(); // Read data from SPI

        // Extracting Information
        debugPrint("Received Value: ");
        debugPrintInt(receivedData);

        // Extract sign_0 (bit 15)
        sign_0 = (receivedData >> 15) & 0x01;

        // Extract sign_1 (bit 14)
        sign_1 = (receivedData >> 14) & 0x01;

        // Extract value_0 (bits 13-8)
        value_0 = (receivedData >> 8) & 0x3F;

        // Extract value_1 (bits 7-2)
        value_1 = (receivedData >> 2) & 0x3F;

        // Check if the two least significant bits are zero
        if ((receivedData & 0x03) != 0)
        {
            debugPrint("Error: Check bits are not zero.");
            break;
        }

        debugPrint("Sign_0: ");
        debugPrintInt(sign_0);

        debugPrint("Sign_1: ");
        debugPrintInt(sign_1);

        debugPrint("Value_0: ");
        debugPrintInt(value_0);

        debugPrint("Value_1: ");
        debugPrintInt(value_1);


        // Limits
        if (value_0 > max_input)
        {
            input_0 = max_input;
        }
        else
        {
            input_0 = value_0;
        }

        if (value_1 > max_input)
        {
            input_1 = max_input;
        }
```

```
            else
            {
                input_1 = value_1;
            }


            // Sign
            if (sign_0)
            {
                input_0 *= -1;
            }

            if (sign_1)
            {
                input_1 *= -1;
            }

            // Move Motor
            debugPrint("Input_0: ");
            debugPrintInt(input_0);
            debugPrint("Input_1: ");
            debugPrintInt(input_1);
            debugPrint("\n");

            MoveSpeed(flat_0-input_0,1023,ID_0);
            CyDelay(1);
            MoveSpeed(flat_1+input_1,1023,ID_1);

        }
    }
 }


/// FUNCTIONS
void debugPrint(const char *str)
{
    UART_Debug_PutString(str);
}

void debugPrintInt(int n)
{
    char string[5];
    sprintf(string, "%d\n", n);
    debugPrint(string);
}

int ReadError(void)
{
    // Bit 7: Reserved
    // Bit 6: Instruction Error - Set to 1 if an undefined instruction is sent or an
action instruction is sent without a Reg_Write instruction.
    // Bit 5: Overload Error - Set to 1 if the specified maximum torque can't control the
applied load.
    // Bit 4: Checksum Error - Set to 1 if the checksum of the instruction packet is
incorrect.
    // Bit 3: Range Error - Set to 1 if the instruction sent is out of the defined range.
    // Bit 2: Overheating Error - Set to 1 if the internal temperature of the Dynamixel
unit is above the operating temperature range as defined in the control table.
    // Bit 1: Angle Limit Error - Set as 1 if the Goal Position is set outside of the
range between CW Angle Limit and CCW Angle Limit.
    // Bit 0: Input Voltage Error - Set to 1 if the voltage is out of the operating
voltage range as defined in the control table.

    int time = 0;
    while ( (UART_GetRxBufferSize() < 6) && (time < TIME_OUT) )
    {
        time = time + 1;
        CyDelay(5);
    }
```

```
    // Read Response
    int error = -1;

    debugPrint("Bf size : ");
    debugPrintInt(UART_GetRxBufferSize());

    if (UART_GetRxBufferSize() > 0)
    {
        uint8_t incoming_Byte = UART_GetByte();

        if (incoming_Byte == 255)
        {
            UART_GetByte(); // Start BYtes
            UART_GetByte(); // ID
            UART_GetByte(); // Length

            error = UART_GetByte(); // Error Byte
            debugPrint("ERROR CHECK: Returned : ");
            debugPrintInt(error);

            if (error & (1 << 6))
            {
                debugPrint("Instruction Error\n");

            }
            if (error & (1 << 5))
            {
                debugPrint("Overload Error\n");

            }
            if (error & (1 << 4))
            {
                debugPrint("Checksum Error\n");

            }
            if (error & (1 << 3))
            {
                debugPrint("Range Error\n");

            }
            if (error & (1 << 2))
            {
                debugPrint("Overheating Error\n");

            }
            if (error & (1 << 1))
            {
                debugPrint("Angle Limit Error\n");

            }
            if (error & 1)
            {
                debugPrint("Input Voltage Error\n");

            }

            return error;
        }
    }
    debugPrint("ERROR CHECK: NO RETURN \n");
    return error;
}

void LED_Control(uint8_t Status, uint8_t ID)
{
    LED_1_Write(1);

    // WRITE DATA
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();
```

```
    uint8_t Checksum = (~(ID + AX_LED_LENGTH + AX_WRITE_DATA + AX_LED + Status)) & 0xFF;
    uint8_t packet[] = {AX_START, AX_START, ID, AX_LED_LENGTH, AX_WRITE_DATA, AX_LED,
Status, Checksum}; // Initialize packet
    UART_PutArray(packet, 8); // Send data through UART

    LED_1_Write(0);
    debugPrint("LED_Control\n");

    if (ERROR_CHECK)
    {
        ReadError();
    }
}


void Set_ID(uint8_t new_ID, uint8_t ID)
{
    LED_1_Write(1);

    // WRITE DATA
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();
    uint8_t Checksum = (~(ID + AX_ID_LENGTH + AX_WRITE_DATA + AX_ID + new_ID)) & 0xFF;
    uint8_t packet[] = {AX_START, AX_START, ID, AX_ID_LENGTH, AX_WRITE_DATA, AX_ID,
new_ID, Checksum}; // Initialize packet
    UART_PutArray(packet, 8); // Send data through UART

    LED_1_Write(0);
    debugPrint("Set_ID: ");
    debugPrintInt(ID);
    debugPrint(" -> ");
    debugPrintInt(new_ID);

    if (ERROR_CHECK)
    {
        ReadError();
    }

}

void Move(uint16_t Position, uint8_t ID)
{
    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    uint8_t Position_H = Position >> 8; // 16 bits - 2 x 8 bits variables
    uint8_t Position_L = Position;

    debugPrint("Position: ");
    debugPrintInt(Position_H);
    debugPrintInt(Position_L);


    const uint8_t length = 9;
    uint8_t packet[length];

    uint8_t Checksum = (~(ID + AX_GOAL_LENGTH + AX_WRITE_DATA + AX_GOAL_POSITION_L +
Position_L + Position_H)) & 0xFF;

    packet[0] = AX_START;
    packet[1] = AX_START;
    packet[2] = ID;
    packet[3] = AX_GOAL_LENGTH;
    packet[4] = AX_WRITE_DATA;
    packet[5] = AX_GOAL_POSITION_L;
    packet[6] = Position_L;
    packet[7] = Position_H;
    packet[8] = Checksum;
```

```
    UART_PutArray(packet, length);

    LED_1_Write(0);
    debugPrint("Move\n");
    if (ERROR_CHECK)
    {
        ReadError();
    }
}

void MoveSpeed(uint16_t Position, uint16_t Speed, uint8_t ID)
{
    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    uint8_t Position_H = Position >> 8;
    uint8_t Position_L = Position; // 16 bits - 2 x 8 bits variables
    uint8_t Speed_H = Speed >> 8;
    uint8_t Speed_L = Speed; // 16 bits - 2 x 8 bits variables

    const uint8_t length = 11;
    uint8_t packet[length];

    uint8_t Checksum = (~(ID + AX_GOAL_SP_LENGTH + AX_WRITE_DATA + AX_GOAL_POSITION_L +
Position_L + Position_H + Speed_L + Speed_H)) & 0xFF;

    packet[0] = AX_START;
    packet[1] = AX_START;
    packet[2] = ID;
    packet[3] = AX_GOAL_SP_LENGTH;
    packet[4] = AX_WRITE_DATA;
    packet[5] = AX_GOAL_POSITION_L;
    packet[6] = Position_L;
    packet[7] = Position_H;
    packet[8] = Speed_L;
    packet[9] = Speed_H;
    packet[10] = Checksum;

    UART_PutArray(packet, length);

    LED_1_Write(0);
    debugPrint("MoveSpeed\n");
    if (ERROR_CHECK)
    {
        ReadError();
    }
}

void TorqueStatus(uint8_t Status, uint8_t ID)
{
    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    const uint8_t length = 8;
    uint8_t packet[length];

    uint8_t Checksum = (~(ID + AX_TORQUE_LENGTH + AX_WRITE_DATA + AX_TORQUE_ENABLE +
Status)) & 0xFF;

    packet[0] = AX_START;
    packet[1] = AX_START;
    packet[2] = ID;
    packet[3] = AX_TORQUE_LENGTH;
    packet[4] = AX_WRITE_DATA;
    packet[5] = AX_TORQUE_ENABLE;
    packet[6] = Status;
    packet[7] = Checksum;
```

```
    UART_PutArray(packet, length);
    LED_1_Write(0);
    debugPrint("TorqueStatus\n");
    if (ERROR_CHECK)
    {
        ReadError();
    }
}

void SetReturnLevel(uint8_t level, uint8_t ID)
{
    // 0: Do not respond to any instructions
    // 1: Respond only to READ_DATA instructions
    // 2: Respond to all instructions

    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    const uint8_t length = 8;
    uint8_t packet[length];

    uint8_t Checksum = (~(ID + AX_SRL_LENGTH + AX_WRITE_DATA + AX_RETURN_LEVEL + level))
& 0xFF;


    packet[0] = AX_START;
       packet[1] = AX_START;
       packet[2] = ID;
       packet[3] = AX_SRL_LENGTH;
       packet[4] = AX_WRITE_DATA;
       packet[5] = AX_RETURN_LEVEL;
       packet[6] = level;
       packet[7] = Checksum;


    UART_PutArray(packet, length);
    LED_1_Write(0);
    debugPrint("SetReturnLevel\n");
    if (ERROR_CHECK)
    {
        ReadError();
    }
}

void reset(uint8_t ID)
{
    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    const uint8_t length = 8;
    uint8_t packet[length];

    uint8_t Checksum = (~(ID + AX_RESET_LENGTH + AX_RESET)) & 0xFF;

       packet[0] = AX_START;
       packet[1] = AX_START;
       packet[2] = ID;
       packet[3] = AX_RESET_LENGTH;
       packet[4] = AX_RESET;
       packet[5] = Checksum;


    UART_PutArray(packet, length);
    LED_1_Write(0);
    debugPrint("reset\n");
    if (ERROR_CHECK)
    {
        ReadError();
```

```
    }
}

void ping(uint8_t ID)
{
    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    const uint8_t length = 6;
    uint8_t packet[length];

    uint8_t Checksum = (~(ID + AX_READ_DATA + AX_PING)) & 0xFF;

       packet[0] = AX_START;
       packet[1] = AX_START;
       packet[2] = ID;
       packet[3] = AX_READ_DATA;
       packet[4] = AX_PING;
       packet[5] = Checksum;

    UART_PutArray(packet, length);
    LED_1_Write(0);
    debugPrint("ping\n");
    if (ERROR_CHECK)
    {
        ReadError();
    }
}

void MoveSpeedRW(uint16_t Position, uint16_t Speed, uint8_t ID)
{
    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    uint8_t Position_H = Position >> 8;
    uint8_t Position_L = Position; // 16 bits - 2 x 8 bits variables
    uint8_t Speed_H = Speed >> 8;
    uint8_t Speed_L = Speed; // 16 bits - 2 x 8 bits variables

    const uint8_t length = 11;
    uint8_t packet[length];

    uint8_t Checksum = (~(ID + AX_GOAL_SP_LENGTH + AX_REG_WRITE + AX_GOAL_POSITION_L +
Position_L + Position_H + Speed_L + Speed_H)) & 0xFF;

    packet[0] = AX_START;
    packet[1] = AX_START;
    packet[2] = ID;
    packet[3] = AX_GOAL_SP_LENGTH;
    packet[4] = AX_REG_WRITE;
    packet[5] = AX_GOAL_POSITION_L;
    packet[6] = Position_L;
    packet[7] = Position_H;
    packet[8] = Speed_L;
    packet[9] = Speed_H;
    packet[10] = Checksum;

    UART_PutArray(packet, length);

    LED_1_Write(0);
    debugPrint("MoveSpeed Register Write\n");
    if (ERROR_CHECK)
    {
        ReadError();
    }
}

void action(void)
```

```
{
    LED_1_Write(1);
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();

    const uint8_t length = 6;
    uint8_t packet[length];

    packet[0] = AX_START;
    packet[1] = AX_START;
    packet[2] = BROADCAST_ID;
    packet[3] = AX_ACTION_LENGTH;
    packet[4] = AX_ACTION;
    packet[5] = AX_ACTION_CHECKSUM;

    UART_PutArray(packet, length);
    LED_1_Write(0);
    debugPrint("action\n");
}


int ReadVoltage(uint8_t ID)
{
    LED_1_Write(1);

    // WRITE DATA
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();
    uint8_t Checksum = (~(ID + AX_VOLT_LENGTH + AX_READ_DATA + AX_PRESENT_VOLTAGE +
AX_BYTE_READ)) & 0xFF;
    uint8_t packet[] = {AX_START, AX_START, ID, AX_VOLT_LENGTH, AX_READ_DATA,
AX_PRESENT_VOLTAGE, AX_BYTE_READ, Checksum}; // Initialize packet
    UART_PutArray(packet, 8); // Send data through UART


    LED_1_Write(0);
    debugPrint("Read Voltage\n");

    // Wait for time out or Recived bytes
    int time = 0;
    while ( (UART_GetRxBufferSize() < 6) && (time < TIME_OUT) )
    {
        time = time + 1;
        CyDelay(5);
    }

    // Read Response
    int voltage = -1;

    debugPrint("Bf size : ");
    debugPrintInt(UART_GetRxBufferSize());

    if (UART_GetRxBufferSize() > 0)
    {
        uint8_t incoming_Byte = UART_GetByte();

        if (incoming_Byte == 255)
        {
            UART_GetByte(); // Start BYtes
            UART_GetByte(); // ID
            UART_GetByte(); // Length

            uint8_t error = UART_GetByte(); // Error Byte
            if (error != 0)
            {
                voltage = error;
                debugPrint("Error Detected");
                return voltage;
            }
```

```
            uint8_t voltage_byte = UART_GetByte();
            voltage = voltage_byte;
        }

    }

    return voltage;
}

int ReadTemperature(uint8_t ID)
{
    LED_1_Write(1);

    // WRITE DATA
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();
    uint8_t Checksum = (~(ID + AX_TEM_LENGTH + AX_READ_DATA + AX_PRESENT_TEMPERATURE +
AX_BYTE_READ)) & 0xFF;
    uint8_t packet[] = {AX_START, AX_START, ID, AX_TEM_LENGTH, AX_READ_DATA,
AX_PRESENT_TEMPERATURE, AX_BYTE_READ, Checksum}; // Initialize packet
    UART_PutArray(packet, 8); // Send data through UART

    LED_1_Write(0);
    debugPrint("Read Temperature\n");

    // Wait for time out or Recived bytes
    int time = 0;
    while ((UART_GetRxBufferSize() < 6) && (time < TIME_OUT))
    {
        time = time + 1;
        CyDelay(5);
    }

    // Read Response
    int temperature = -1;

    debugPrint("Buffer size: ");
    debugPrintInt(UART_GetRxBufferSize());

    if (UART_GetRxBufferSize() > 0)
    {
        uint8_t incoming_Byte = UART_GetByte();

        if (incoming_Byte == 255)
        {
            UART_GetByte(); // Start Bytes
            UART_GetByte(); // ID
            UART_GetByte(); // Length

            uint8_t error = UART_GetByte(); // Error Byte
            if (error != 0)
            {
                temperature = error;
                debugPrint("Error Detected");
                return temperature;
            }

            uint8_t temperature_byte = UART_GetByte();
            temperature = temperature_byte;
        }
    }

    return temperature;
}

int ReadPosition(uint8_t ID)
{
    LED_1_Write(1);

    // WRITE DATA
```

```
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();
    uint8_t Checksum = (~(ID + AX_POS_LENGTH + AX_READ_DATA + AX_PRESENT_POSITION_L +
AX_BYTE_READ_POS)) & 0xFF;
    uint8_t packet[] = {AX_START, AX_START, ID, AX_POS_LENGTH, AX_READ_DATA,
AX_PRESENT_POSITION_L, AX_BYTE_READ_SPEED, Checksum}; // Initialize packet
    UART_PutArray(packet, 8); // Send data through UART

    LED_1_Write(0);
    debugPrint("Read Position\n");

    // Wait for time out or Recived bytes
    int time = 0;
    while ((UART_GetRxBufferSize() < 7) && (time < TIME_OUT))
    {
        time = time + 1;
        CyDelay(5);
    }

    // Read Response
    int position = -1;

    debugPrint("Buffer size: ");
    debugPrintInt(UART_GetRxBufferSize());

    if (UART_GetRxBufferSize() > 0)
    {
        uint8_t incoming_Byte = UART_GetByte();

        if (incoming_Byte == 255)
        {
            UART_GetByte(); // Start Bytes
            UART_GetByte(); // ID
            UART_GetByte(); // Length

            int error = UART_GetByte(); // Error Byte
            if (error != 0)
            {
                position = error;
                debugPrint("Error Detected");
                return position;
            }

            uint8_t position_low_byte = UART_GetByte(); // Position Low Byte
            uint8_t position_high_byte = UART_GetByte(); // Position High Byte
            position = (position_high_byte << 8) + position_low_byte;
        }
    }

    return position;
}

int ReadSpeed(uint8_t ID)
{
    LED_1_Write(1);

    // WRITE DATA
    UART_ClearRxBuffer();
    UART_ClearTxBuffer();
    uint8_t Checksum = (~(ID + AX_POS_LENGTH + AX_READ_DATA + AX_PRESENT_SPEED_L +
AX_BYTE_READ_SPEED)) & 0xFF;
    uint8_t packet[] = {AX_START, AX_START, ID, AX_SPEED_LENGTH, AX_READ_DATA,
AX_PRESENT_SPEED_L, AX_BYTE_READ_SPEED, Checksum}; // Initialize packet
    UART_PutArray(packet, 8); // Send data through UART

    LED_1_Write(0);
    debugPrint("Read Speed\n");

    // Wait for time out or Recived bytes
    int time = 0;
```

```
    while ((UART_GetRxBufferSize() < 7) && (time < TIME_OUT))
    {
        time = time + 1;
        CyDelay(5);
    }

    // Read Response
    int speed = -1;

    debugPrint("Buffer size: ");
    debugPrintInt(UART_GetRxBufferSize());

    if (UART_GetRxBufferSize() > 0)
    {
        uint8_t incoming_Byte = UART_GetByte();

        if (incoming_Byte == 255)
        {
            UART_GetByte(); // Start Bytes
            UART_GetByte(); // ID
            UART_GetByte(); // Length

            int error = UART_GetByte(); // Error Byte
            if (error != 0)
            {
                speed = error;
                debugPrint("Error Detected");
                return speed;
            }

            uint8_t speed_low_byte = UART_GetByte(); // Position Low Byte
            uint8_t speed_high_byte = UART_GetByte(); // Position High Byte
            speed = (speed_high_byte << 8) + speed_low_byte;
        }
    }

    return speed;
}
```

11.7 Appendix G: Engineering Drawings

This section contains the CAD drawings for the assembly and all components of the project.