
# EG2310

# Group 3

# G1

# Report

Student Team Members:

| | |
|---|---|
| Tan Ping Hui | A0272169X |
| David Michael Indraputra | A0276057X |
| Peteti Harshith | A0288271W |
| Song Cheng Yan | A0276267R |
| Basudeb Chakraborty | A0288113E |

# Table of Contents

# 1. Problem Definition

## 1.1 Mapping

The TurtleBot must perform mapping operations to delineate maze elements and generate a comprehensive Simultaneous Localization and Mapping (SLAM) map of the designated area with full map closure.

## 1.2 Navigation

The TurtleBot initiates its trajectory from a designated starting point and proceeds towards a maze with random maze elements, in which the TurtleBot needs to navigate through and find the exit.

## 1.3 Enter Locked Room

After clearing the maze, the TurtleBot will encounter two rooms with a door equipped with an electric lock that unlocks via a web server. To unlock either door, the TurtleBot will need to initiate a HyperText Transfer Protocol (HTTP) call to the server. If the request is valid, the designated door will be unlocked, and the HTTP response will reply with the ID of the unlocked door. The Turtlebot then needs to navigate towards the unlocked room.

## 1.4 Launching Ping Pong into a Bucket

Upon entering the unlocked room, the TurtleBot needs to locate a bucket and project five ping-pong balls into it.

## 1.5 Optional Markers

The mission can be assisted by strategically placing temporary markers before the initiation of the mission. These markers serve as reference points, facilitating the identification of distinct zones and points of interest within the mapped environment.

# 2. Literature Review

## 2.1. Mapping

The Turtlebot has an integrated 360 Light Detection and Ranging (LiDAR) sensor known as LDS-02. This sensor employs pulsed lasers to measure distances in a 360° range, enabling SLAM as well as navigation.

### 2.1.1 Mapping Algorithms

GMapping
GMapping creates a grid map of the environment while simultaneously pinpointing the TurtleBot's location. It uses sensor data to update the map's occupancy probabilities and the TurtleBot's pose, employing a probabilistic framework like the Rao-Blackwellized Particle Filter. This process involves prediction, measurement update, particle resampling, map update, and loop closure to refine both the map and the TurtleBot's position iteratively.

Cartographer
Cartographer employs scan matching and loop closure techniques to map the environment and localise the TurtleBot. It optimises pose estimates and map consistency, integrating sensor data for real-time operation in various robotic applications.

## 2.2 Navigation

### 2.2.1. Line Following using IR Sensors

The system utilises two separate infrared (IR) sensors positioned at a set distance from each other. By strategically placing black tapes throughout the maze beforehand, the TurtleBot can follow the tape by adjusting left or right based on the minimal reflected IR detected by either sensor.

### 2.2.2 Path Planning Algorithms

Navigation algorithms can be used to identify the most efficient route to clear the maze while recognising and avoiding potential obstacles during its movement.

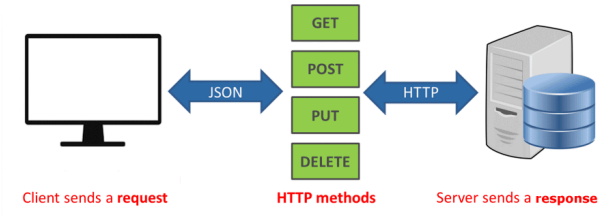
A* Search Algorithm
A* is a graph-based search algorithm that systematically explores the search space by evaluating nodes based on a combination of the cost to reach them from the start node and an estimated cost to reach the goal from them. It uses heuristics to guide the search towards the goal efficiently.

Rapidly exploring random tree (RRT)
RRT is a sampling-based algorithm for path planning. It incrementally grows a tree structure rooted at the starting configuration, exploring the configuration space by randomly sampling new nodes and connecting them to the nearest node in the tree.

## 2.3. Enter Locked Room

### 2.3.1. HTTP Call

HTTP, an application layer protocol, facilitates information exchange between networked devices like the TurtleBot and an electric lock system integrated with a web server capable of receiving and responding to requests. These requests, containing a URL and an HTTP method (such as POST or GET), dictate the action to be executed by the server. For unlocking doors, a POST request is utilised. The request's URL directs to a specific endpoint on the server responsible for door control operations, like "https://your-door-server.com/unlock". Upon initiation, the web server processes the request, interprets the provided data, and coordinates with the integrated electric lock system. This coordination involves sending commands or signals to release the door lock mechanism upon validation. Subsequently, one of the two doors unlocks, with the server transmitting a signal containing the ID of the unlocked door back to the TurtleBot.




### 2.3.2. Locate Unlocked Room

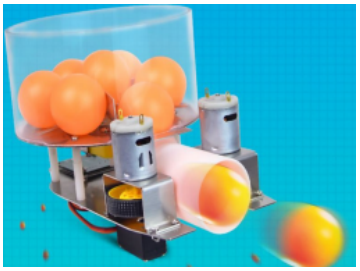
Line Sensors

After traversing the maze, the TurtleBot will encounter a T-shaped configuration of black tape, causing both infrared (IR) sensors to receive minimal reflected IR waves. Sensing this pattern, the TurtleBot halts its movement and initiates an HTTP call. The response to this call provides the ID of the unlocked door. Depending on this information, the TurtleBot turns either left or right, guiding it through the unlocked door.

## 2.4. Launching Ping Pong

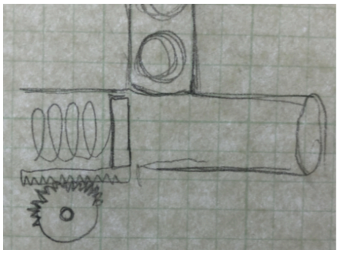
### 2.4.1 Flywheel Launcher

A flywheel launcher employs high-speed, low-torque motors to spin its wheels rapidly. This innovative design comes in one- and two-wheeled configurations, both adept at accelerating a ping-pong ball as it traverses the whirring wheels. Subsequently, the propelled ball is ejected with precision.



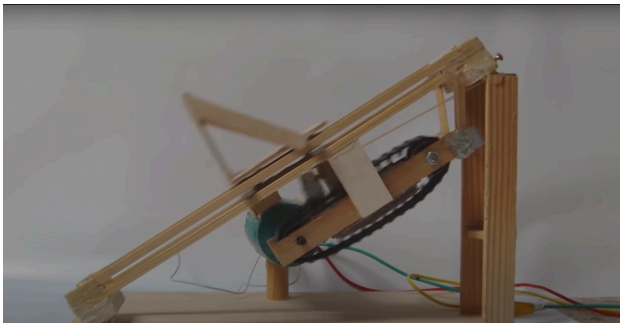
### 2.4.2 Spring Launcher

A spring launcher operates by compressing a spring to store potential energy, which is then released to propel an object forward using the converted kinetic energy. By controlling the amount of compression, the launch force can be adjusted to achieve the desired velocity and trajectory for the ping pong ball.



### 2.4.3 Slingshot Launcher

A slingshot ping pong launcher functions by employing elastic bands or rubber tubing stretched between fixed points on a sturdy frame to store energy. This tension is released through a firing mechanism, typically triggered by pulling a lever or string, which propels the ping pong ball forward. Before launch, the ball is securely held in place by a mechanism such as a fabric pouch or cup-like holder. The tension and launch angle can be customised for different trajectory and distance.



# 3. Concept Design

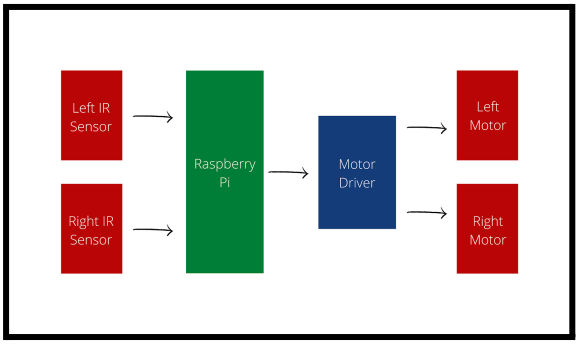
## 3.1 Line Following System

### 3.1.1 Infrared Sensor (TCRT5000)

Line navigation will employ IR Sensors positioned at each TurtlrBot's end. These sensors comprise a transmitter, responsible for emitting signals, and a receiver, which detects the emitted signals. When the IR sensors emit infrared light onto an object, light absorbed by black surfaces results in low output. In contrast, light reflected by white surfaces returns to the transmitter, detected by the infrared receiver, yielding an analogue output.

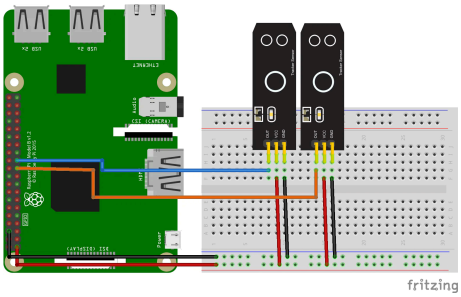
### 3.1.2 Software

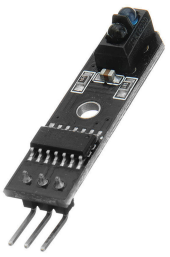
The R-Pi processes input from the IR sensors to control the TurtleBot's motors. When the TurtleBot veers off-course to the left, the right IR sensor detects the black tape, generating a low output. In response, the R-Pi decreases the right wheel's speed, nudging the TurtleBot back on track to the right. The process is reversed for deviations to the right. Upon encountering a T-shaped configuration of black tape, both IR sensors detect it, prompting the R-Pi to halt the wheels completely and prepare for the next command.  Detailed descriptions of these scenarios are provided in Appendix A. An example of the code is also provided in Appendix B.




### 3.1.3 Electrical

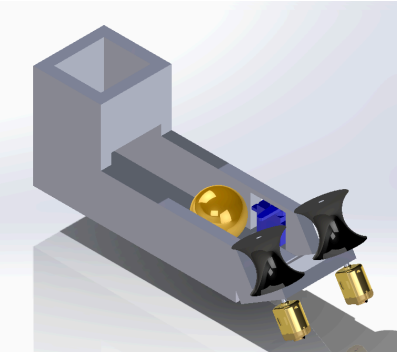
The output pins of the IR sensors will be connected to the GPIO pins on the R-Pi for reading their signals. The IR sensors, TCRT5000, operate at a voltage range of 5V ± 0.25V (Appendix C), which is supported by the output of the R-Pi.

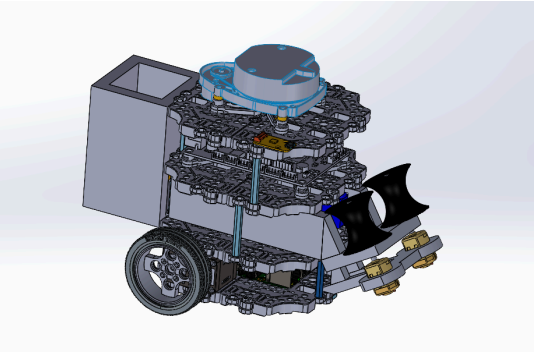


### 3.2 Flywheel Launcher System

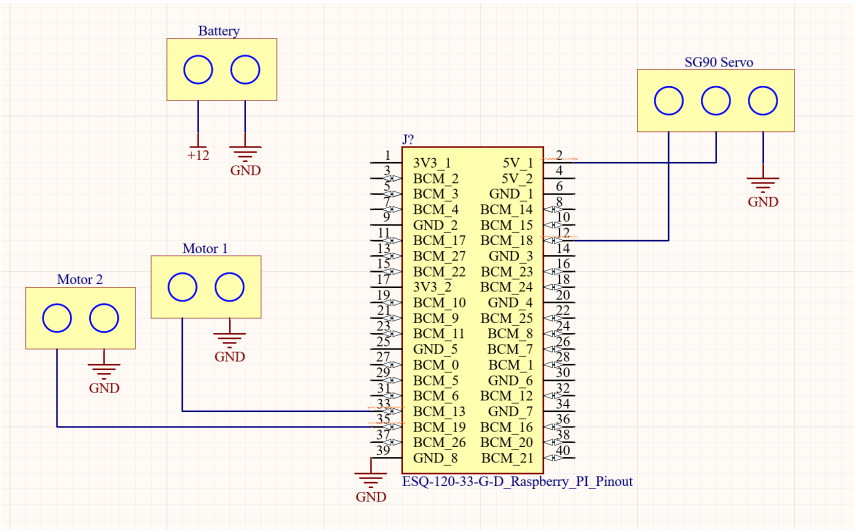
#### 3.2.1. Mechanical





The design incorporates a high-speed flywheel mechanism, which is designed to be able to accomplish the goal of shooting five balls continuously into a pail 50 cm away. It utilises two 20000 rpm motors that can reach the needs of 17000 rpm. The system stores balls in the second layer of turtlebot and a carriage behind it. When the turtlebot reaches the shooting range, the servo will release the balls after the acceleration of the flywheels. This design intends to keep the centre of gravity in the middle so as not to decrease too much speed and avoid having the carriage blocking the lidar when mapping. The figures above show the CAD model of the prototype. More detailed CAD drawing and calculation is provided in Appendix D and Appendix E respectively

#### 3.2.2 Electrical




#### 3.2.3 Software

Upon precise positioning of the TurtleBot and aligned with the bucket, the R-Pi will activate the motor to launch all 5 balls continuously, which will be powered by a predetermined voltage for a timed duration. With the completion of this task, all mission objectives are achieved, prompting the automatic cessation of all program operations.

# 4. BOGAT

## 4.1 Mapping

| GMapping | Cartographer |
|---|---|
| -Less computationally intensive<br>-Easier to configure | -Multi-sensor support<br>-Better performance in highly dynamic environments<br>-Higher accuracy with loop closure detection and scan matching |
| Decision: Considering that the mission maze is small and contains non-moving elements, GMapping is chosen due to its simplicity, efficiency, and suitability for static environments. It provides a reliable solution for mapping and localisation tasks while maintaining compatibility with the TurtleBot platform. | |

## 4.2 Navigation

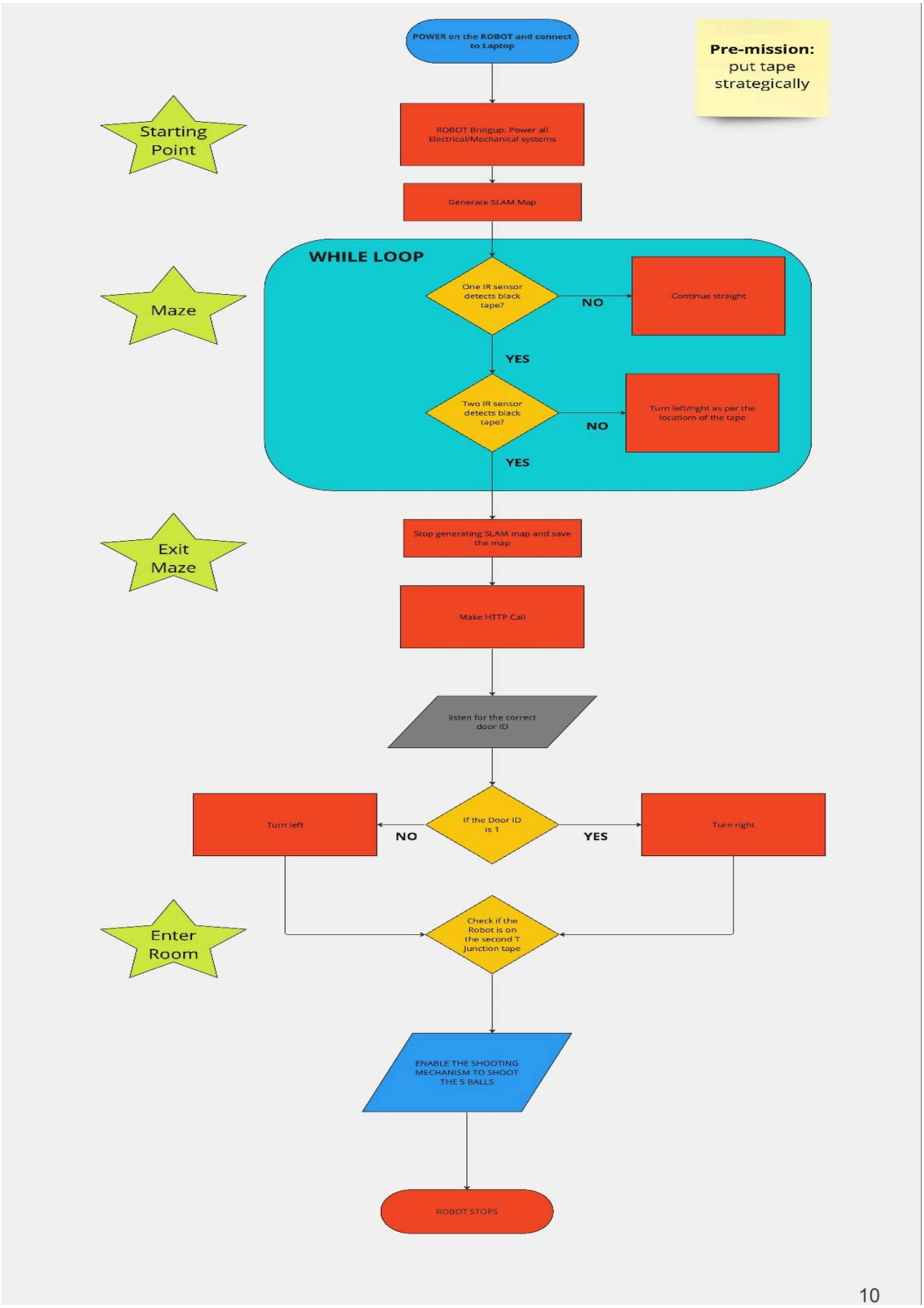
| Line Following | A* Search Algorithm | Rapidly exploring random tree |
|---|---|---|
| -Heavily dependent on configuration of tapes placed before mission<br>-Requires minimal space and computational power | -Guarantees the shortest path<br>-High space and computational complexity | -Guarantees a working solution<br>-Can handle partially known environments<br>-Easy to configure<br>-Can work in dynamic environments |
| Decision: Line following offers a straightforward and cost-effective navigation method compared to path-following algorithms, which are more complex and computationally intensive. Line following also excels in controlled environments with clear line markings, providing robust and reliable navigation. The simplicity of implementation also makes it more efficient and predictable. | | |

## 4.3 Launching Ping Pong

| Flywheel | Spring | Slingshot |
|---|---|---|
| -Rapid firing<br>-Adjustable speed<br>-High power consumption<br>-Complex design | -Energy efficient<br>-Compact design<br>-Limited firing range<br>-Difficult to manufacture | -Energy efficient<br>-Complex design to load autonomously<br>-Less accuracy |
| Decision: Despite the flywheel mechanism's high power consumption, its rapid firing provides a significant advantage in expediting mission completion. Additionally, its adjustable speed ensures a consistent firing trajectory compared to the other designs. | | |

# 5. Preliminary Design

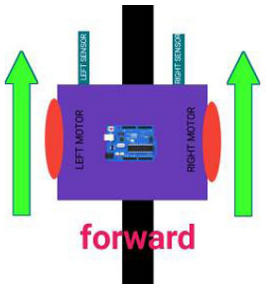
## 5.1 Mission Plan



# Appendix A - Line Following Principle

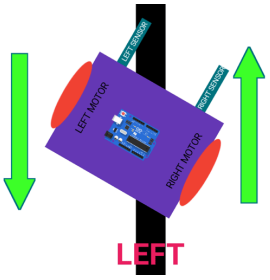
**Moving Forward**

In this scenario, when both sensors detect a white surface and the line is situated between them, the TurtleBot should proceed forward. This means both motors should rotate in a manner that propels the TurtleBot forward.



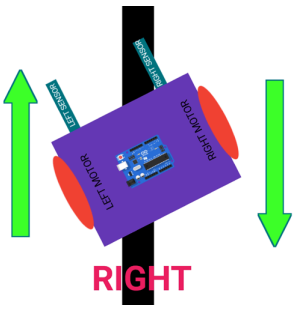

**Turning left**

In this situation, the left sensor detects the dark line while the right sensor registers the white surface. Consequently, the microcontroller receives a signal from the left sensor indicating the presence of the black line. As a result, the TurtleBot is instructed to turn left. Thus, the left motor rotates in reverse while the right motor moves forward, causing the TurtleBot to turn leftward.



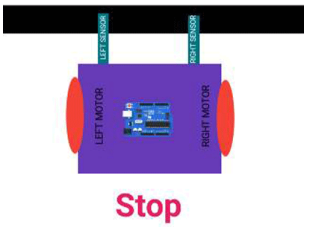

**Turning Right**

In this scenario, similar to the previous case, only the right sensor detects the line, indicating that the TurtleBot should turn right. To execute a right turn, the left motor rotates forward while the right motor rotates backward. Consequently, the TurtleBot pivots in the right direction.




**Stopping**

In this particular situation, both sensors detect the black line simultaneously, prompting the R-Pi to interpret this condition as a command to halt. Consequently, both motors are stopped, making the TurtleBot reach a standstill.



# Appendix B - Line Following Code

```
def control_motors():

    # Read sensor inputs

    left_sensor_value = GPIO.input(left_sensor_pin)

    right_sensor_value = GPIO.input(right_sensor_pin)


    # Adjust motor speeds based on sensor inputs

    if left_sensor_value == 0:  # Black tape detected on left sensor

        GPIO.output(left_motor_pin, GPIO.LOW)  # Stop left motor

        GPIO.output(right_motor_pin, GPIO.HIGH)  # Move right motor forward

    elif right_sensor_value == 0:  # Black tape detected on right sensor

        GPIO.output(left_motor_pin, GPIO.HIGH)  # Move left motor forward

        GPIO.output(right_motor_pin, GPIO.LOW)  # Stop right motor

    else:  # No black tape detected

        GPIO.output(left_motor_pin, GPIO.HIGH)  # Move both motors forward

        GPIO.output(right_motor_pin, GPIO.HIGH)
```

# Appendix C - Datasheet

| Component | Datasheet |
|---|---|
| TCRT5000 | https://www.utmel.com/pdf/datasheets?PdfFile=r/datasheets/vishaysemiconductoroptodivision-tcrt5000l-datasheets-4954.pdf&product=vishaysemiconductoroptodivision-tcrt5000l-6029927 |
| Micro Electric 030 Motor FF-030PB-08315 | https://shopee.sg/product/925606537/24402422194?gad_source=1&gclid=Cj0KCQiAn-2tBhDVARIsAGmStVm2RJYpclP2GBFBQBEqS3HhQgqTroiLIPNdyVbCQmPV0BijyRJHF4AaApUXEALw_wcB |
| Servo SG90 | http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf |

# Appendix D - CAD Model

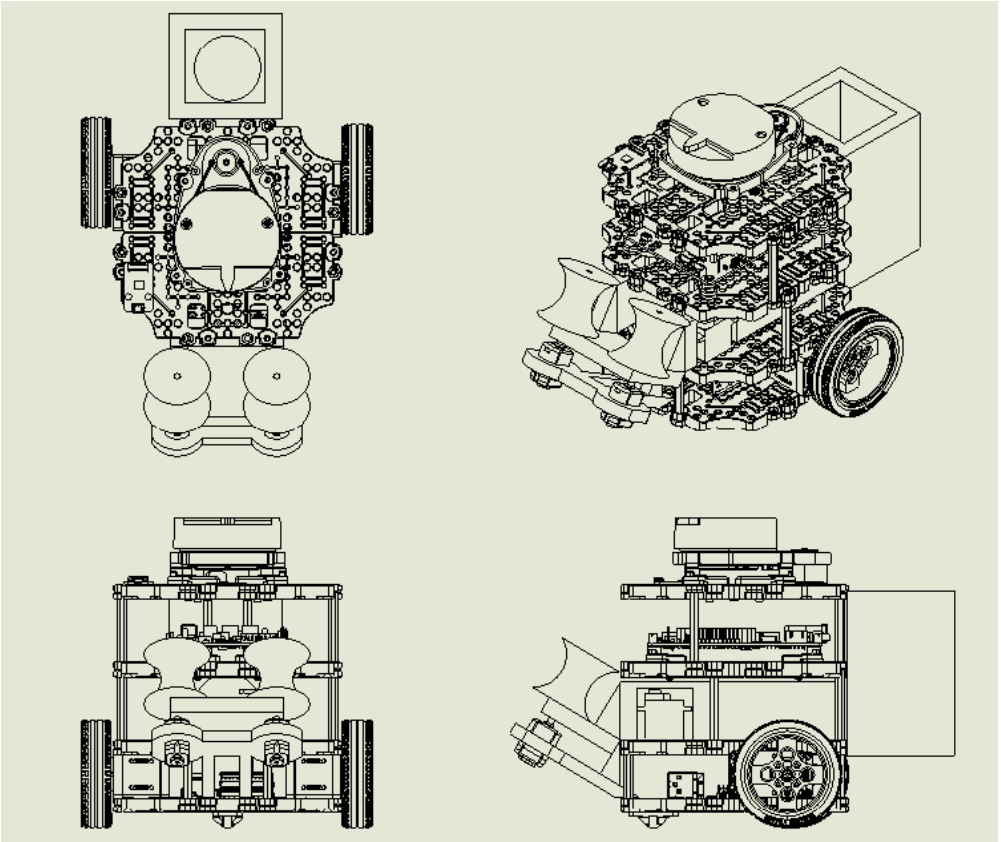


Figure 1: CAD Drawing of the Turtlebot

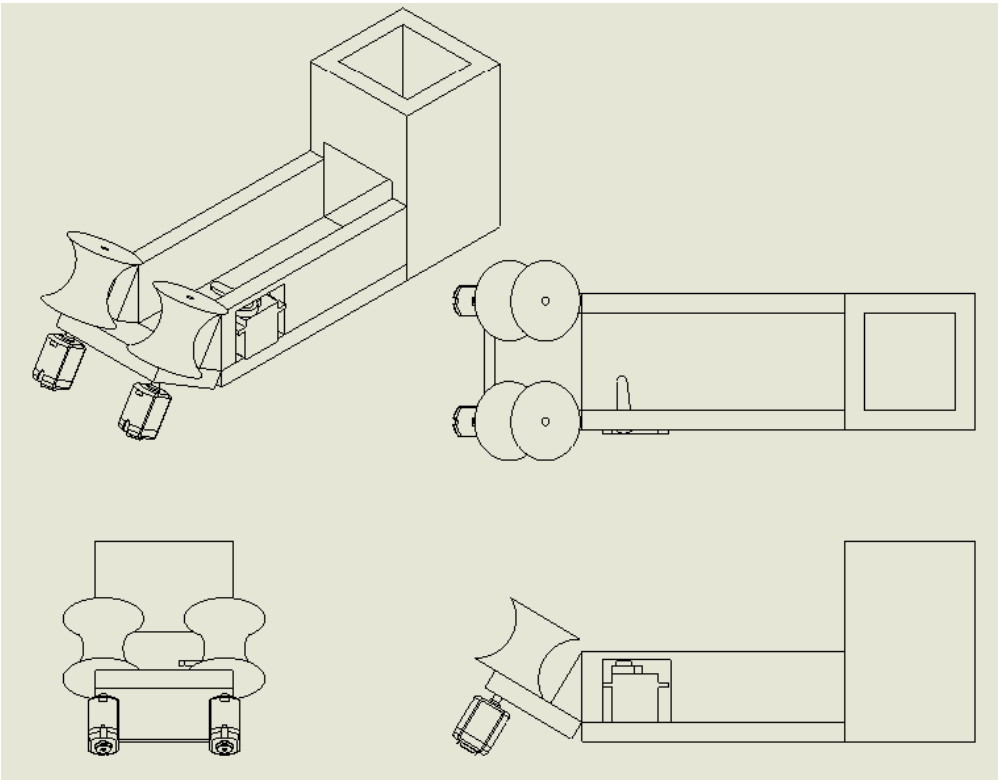


Figure 2: CAD Drawing of the Payload

# Appendix E - Calculation for Launching Ping Pong

For the ball to shoot into a 35cm height bucket, both flywheels' motors need to be at least 17000rpm, and the track needs to be tilted up by 32 degrees, as shown in the calculations below. However, to account for air resistance, which was neglected in the calculations, we chose motors that can run up to 20000rpm.