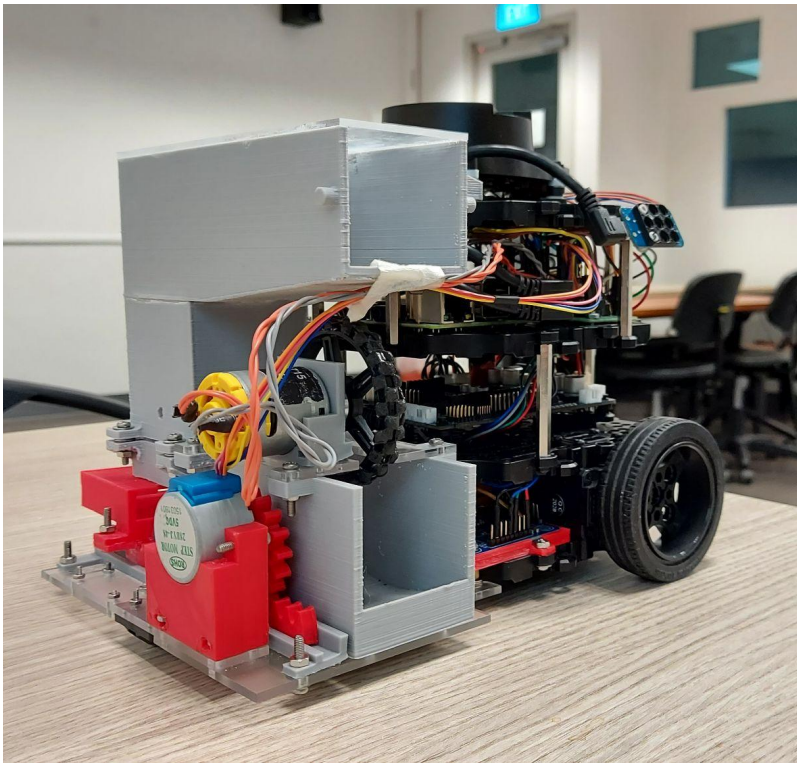
EG2310: Fundamentals of System Design
Final Report



Group 8

Atreyee Basak (A0237089U)
Ding Yitian (A0238472Y)
Tan Chern Lin Justin (A0240108W)
Teoh Xu En (A0239559L)

Table of Contents

Introduction

Throughout this project, the project problem was tackled with the approach of a V-Model. The flow of the report will mirror that of the V-Model as shown below.

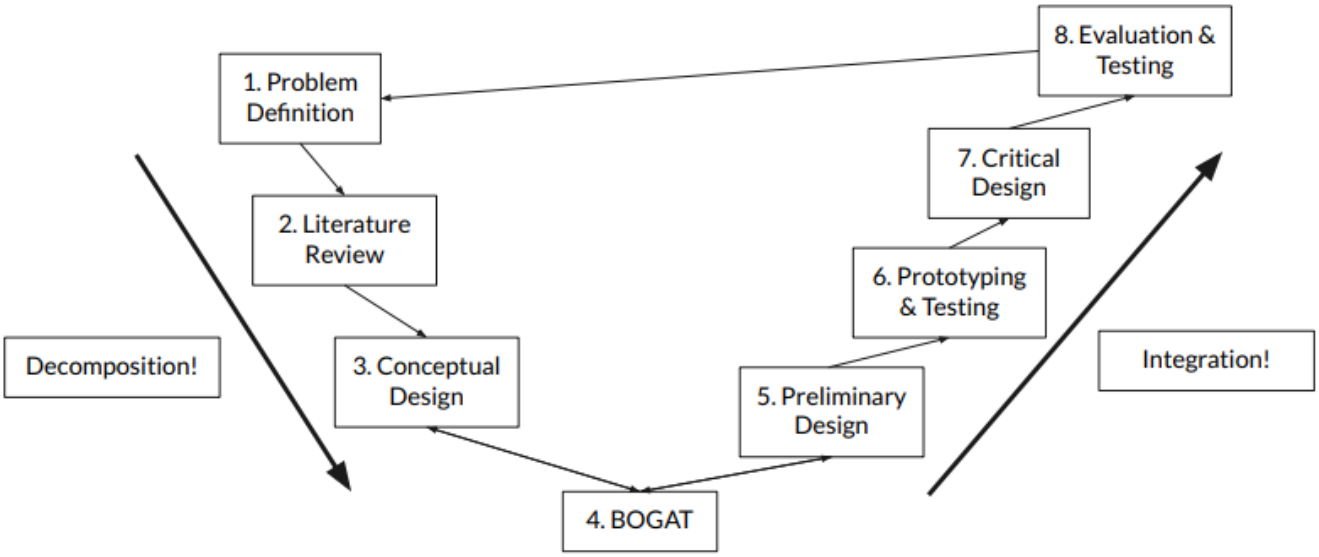


Figure: V-Model used in EG2310

Stage 1: Problem Definition

Definition of Stakeholder Expectations

To fabricate a Turtlebot Burger robot capable of autonomously navigating a maze confined to dimensions of 5m by 5m. The pathway will start from a designated ball loading zone verified via NFC tags located on the wall and on the floor. The TA will manually load up to 5 ping pong balls while the robot remains static. Upon loading, the robot will navigate the maze to a biscuit tin that emits an infrared signature. The tin can will be placed on the floor and there is no restriction in terms of the distance between the robot and the tin can while firing. The robot is required to detect and shoot at least 1 ping pong ball at the hot target successfully. These tasks must be done concurrently in one attempt.

Technical requirements

Requirement Number	**"Shall" Statement**	**Verification Success Criteria**	**Verification Method**
1.1	The robot should recognise the right direction to move and move in that direction.	The robot must manoeuvre within the empty spaces without hitting the walls of the maze.	Given map data from LiDAR sensors, it should manoeuvre around a randomly generated maze autonomously.
1.2	The robot should stop and remain in position upon detecting the NFC label(s).	The robot must stop moving more than 5cm upon detecting NFC label	Place NFC labels in the vicinity and make sure that when the robot goes near it, it will stop moving.
1.3	The robot shall wait in position till further interaction by the TA	The robot must not move off during loading and should begin to move upon being prompted by the TA.	Place ping pong balls into the storage/ launching mechanism one by one and check if it moves off.
2.1	The robot should map out the maze using the LiDAR sensor.	Maps reconstructed by LiDAR sensors must be similar to the actual maze.	The robot shall navigate through multiple randomly generated mazes and LiDAR sensors should map out mazes accurately.
2.2	The robot should be able to acquire data from IMU	The robot must correctly identify its orientation.	Place the robot on a piece of paper marked at 45° intervals with respect to the centre. Manually rotate the robot and check if the robot gives the correct output.

2.3	The robot should acquire data from the NFC tag.	Upon entering the range of NFC tags, the robot is able to receive and interpret the data from it.	A signal is emitted and the Raspberry Pi computer should receive data from the sensor when the sensor is placed near the NFC tag.
2.4	The robot should acquire and process data from thermal sensor	The robot should be able to receive data from the thermal sensor and recognise it as a heat signature and mark its location	When the robot is place near a heat signature it should be able to identify its location
3.1	The robot shall be able to store five ping pong balls.	The robot should have storage for five ping pong balls.	NA
4.1	The robot should aim precisely towards the heated target.	The robot should be able to aim and align the trajectory of the ping pong ball with the target.	The robot will identify the target and align and aim at the target.
4.2	The robot should propel the ping pong ball towards the correctly identified target.	Ping pong ball should collide with the target upon firing.	Ping Pong ball will be fired after alignment to target.
5.1	The robot shall have sufficient power to sustain electrical modules onboard while travelling.	The robot shall be able to sustain for 30 minutes.	The robot will be equipped with other modules and put through rigorous endurance testing, covering an area of $50m^2$.

Stage 2: Literature Review/Research

2.1 Navigation

Simultaneous Localisation and Mapping (SLAM) [1] uses data from the odometer, Laser Distance Sensor (LDS) and other sensors to understand its environment. Gmapping and Cartographer from ROS can generate a map of the environment using two-dimensional Occupancy Grid Map (OGM) [2], where the white area is collision-free, black area is occupied and grey represents the unknown area. Different SLAM and path planning algorithms can be tested on the Gazebo Simulator.

There are many algorithms [7] on the topic of SLAM. For known environment algorithms, there are Lee's algorithm, dead-end filling algorithm, maze-router algorithm, soukup algorithm, Hadlock algorithm, flood-fill algorithm and heuristic search algorithm. For unknown environment algorithms, there are wall-follower, tremaux, random mouse and pledge algorithms. Autonomous maze solving robotic systems usually depend on camera-based systems or sensor-based systems. The team can rapidly test these algorithms on the Gazebo Simulator.

2.2 Sensors

Near-field communication (NFC) technology [5] uses electromagnetic radiation for transfer of data. The ADAfruit PN532 NFC/RFID Controller Shield for Arduino [20] is used for 13.56MHz NFC devices and contains a PN532 chip-set. It has a range of 10cm and cannot be hidden behind a metal barrier. It uses I2C communication protocol. If it reads an NFC signal, it will "interrupt" the program automatically. It is designed to be used with a 3.3V system, a logic level converter is required to work with a 5V system.

The Lidar Sensor on Turtlebot [6] is able to sense 360 degrees to collect data from SLAM. It may be used for 3D sensing, self parking and SLAM. It has a sampling rate of 1.8kHz and a detection distance of 120mm ~ 3500mm. It consumes 400mA or less of current and its light source is the semiconductor laser diode.

The thermal sensor works by measuring non-contact radiation. The Thermal Sensor AMG8833 [9] operates at 3.3V to 5V, at a sample rate of 1Hz to 10Hz. This sensor contains an 8X8 array of IR thermal sensors.

2.3 Firing Mechanism

A cam [11] is used for translating rotary motion into linear motion. The follower is connected to the cam's motion and travels usually in a linear fashion. A cam usually consists of a rotating shaft that has an irregular or oblong shape. An external force (e.g. spring) is usually required to maintain motion. Such a mechanism can be refitted for the purpose of firing ping pong balls with an automatic reloading system.

An axle is mounted with cams and passes through a piston connected to springs [27]. Cams are connected to a motor which will drive the springs back. Springs are more durable and a barrel will increase accuracy.

Flywheel [13][14] is a mechanism designed to store rotational energy as efficiently and possible. It has high consistency and accuracy. It features one or many wheels turning at high speeds with a ball compressed against them. To calculate the trajectory of the ball [16] the team can look at the different forces acting on the ball to calculate the velocity and spin.

Taking reference from a Nerf gun [15] a piston can be connected to a slider. Pulling the trigger will disconnect the piston and elastic energy from the compression of the spring powers the piston. This in turn pushes the bullet forward.

The catapult [17] relies on rotational movement in order to fire objects. This is done by swinging a catapult arm with an object attached to the end and stopping it abruptly, while the object gains momentum and continues on the trajectory. While an elastic catapult is easy to tune by changing the amount of rubber bands, it places a large amount of stress on the robot due to its tension. Additionally, the short lifespan of the rubber bands might lead to failure during its mission. The team can also use slip gear as a trigger mechanism for a catapult, where a portion of a gear is shaven off. Other ways to power a catapult are nautilus gears/cams and connecting rods.

2.4 Other/General

Looking at videos of past runs [21]. The team noticed that seniors have placed the launching mechanism at different places, from the top, to the sides and at the back. If placed at the back, there can be a supporting platform connected to extra wheels for additional stability. Additionally, the location of the Raspberry Pi might be shifted to accommodate more space for other mechanisms.

The Main controller of turtlebot is the OpenCR[6] and it supports 3.3V, 5V and 12V power outputs for the SBC and sensors. It contains a 3 axis Gyroscope, 3 axis accelerometer and a 3 axis magnetometer (MPU-9250). There are 4 programmable LEDs and 2 user buttons. Additionally a 360 Laser Distance Sensor LDS-01 is mounted on the turtlebot [22]. Dynamixel XL430-W250-T [22] are used as Turtlebot motors. The battery used is a 11.1V lithium polymer battery with a capacity of 1800mAh [22].

Stage 3: Concept Design

1.0 Systems Overview

A payload will be attached to a modified Turtlebot Burger to store and fire ping pong balls at a heated tin can. The robot will observe its environment through sensors like: LiDAR and thermal sensors. The data will be processed on the Raspberry Pi and used to traverse the maze and locate the heated can. A firing mechanism will then be used to aim and fire the loaded balls towards the heated tin.

2.0 Documents

The project is described in the file titled "(G1) Engineering Design Principles - 13 Jan 2022.pdf" on slide 22. The project phases deadline are listed in the file title "(G0.5) EG2310 Introduction - Jan 2022.pdf" on slide 11 and 12.

3.0 Description of Envisioned System

3.1 Overview of Mechanical Section

3.1.1 Firing Mechanism

3.1.1.a Flywheel

Upon researching firing mechanisms, The team encountered a video of a flywheel system [13] shooting out paper aeroplanes using lego wheels. The team could adapt such a system to fire ping pong balls instead by adjusting the gap between the wheels. This system comprises one motor and a gearbox with two wheels spinning in the opposite directions. A sketch of the idea can be found in Appendix A Figure 1.

3.1.1.b Solenoid

A solenoid can be used to impart a linear force on the ball, 'firing' it [24][25]. Using a mechanical relay or MOSFET, the team can close the circuit momentarily to generate a magnetic field that will exert force on the piston. A sketch of the idea can be found in Appendix A Figure 2.

3.1.1.c Catapult Arm

The object will be placed on a launching cradle connected to a launching arm which will move upwards and outwards about a pivot point. Using a servo to generate the angular force about the pivot, the ball will be launched through the air and towards the target [26]. A sketch of the idea can be found in Appendix A Figure 3.

3.1.1.e Cam Based Shooting Mechanism [27]

An oblong shaped cam will be used to pull back the piston which is anchored to a spring. The cam is connected to gears travelling in circular motion which will start turning when the target is identified and a signal to fire is sent to the motor. The oblong shaped cam will have a dented area, which when reached, will release the piston. The piston will jerk forward, pushing the ball out. A sketch of the idea can be found in Appendix A Figure 4.

3.1.2 Storage System

3.1.2.a Vertical Column

Taking inspiration from a bullet magazine, the team found that a vertical column of storage used with a solenoid will allow us to fire ping pong balls the robot has stored. The vertical column will use gravity to move the ball from the storage section to the firing mechanism. A sketch of the idea can be found in Appendix A Figure 5 and 7.

3.1.2.b Carousel

Upon reviewing the previous group's robot design for this robot, the team came across a carousel storage system mounted on the top layer of the robot. The carousel feeds the ping pong balls into the firing mechanism by rotating the whole storage. A sketch of the idea can be found in Appendix A Figure 6.

3.1.3 Robot Structure

The additional storage or launching features, such as the launching and storage system will be added to the Turtlebot Burger in the form of an extension, hence, the structure of the current Turtlebot will not change. The team plans to mount the storage and firing mechanism by the side of the robot. These systems will be supported with mounting structures that are anchored on the plates of the Turtlebot. Due to the limited space on the robot, the team will be mounting the circuit board connecting the different sensors on the back of the robot, with a piece of acrylic plate shielding it from possible damages.

3.2 Overview of Electrical Section

3.2.1 Sensors

3.2.1.a Thermal Camera

To identify a heated tin, a thermal sensor [9][28] has to be used. An 8 x 8 thermal camera allows for better targeting as the location of the heat source can be localised to an area.

3.2.1.b NFC Sensor

An NFC Sensor will allow the robot to detect the presence of NFC tags [29]. The robot will then stop for the loading of the balls. It is to be placed under the robot due to its limited range.

3.2.1.c LiDAR

A LiDAR [30] [31] sensor is mounted on the top of the Turtlebot. It is a precise, accurate and flexible method to plot the surroundings. It will allow the robot to map the environment and route to travel, preventing it from hitting walls and other obstacles. The sensor used in the robot is 360 Laser Distance Sensor LDS-01.

3.2.1.d Inertial Measurement Unit (IMU)

A 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer are onboard the robot main controller board. These sensors provide orientation data, allowing the slam algorithm to have information regarding the robot's heading relative to the environment.

3.2.2 Power

The power demand will determine how long the robot and the functions it performs can be sustained. The robot is powered by a 11.1V 1800mAh LiPo battery that is expected to last for 2.5 hours.

3.2.3 Robot-Human interface

An array of buttons will be used to allow external interactions with the robot. Functions like indicating the end of the ball loading phase to begin movement again, starting the robot and shutting down the Raspberry Pi can be programmed into these buttons.

3.3 Overview of Software Section

3.3.1 ROS

Robot Operating System [32][33] allows the program to be modular. The nodes controlling the different sections of the robot can communicate with one another via messages published on to topics. In this case the sensor data can be published on to topics for a separate node to process.

3.3.2 Mapping of Maze

The Simultaneous Localisation and Mapping (SLAM) [34] technique is used to draw a map based on data from the LiDAR sensor and odometry information. The robot travels around the maze and marks empty spaces as white, black area as inaccessible and grey area as unknown. The map uses the two-dimensional Occupancy Grid Map (OGM) to map the area. The maze is mapped so as to allow the robot to navigate the maze. The robot will move in the direction of the unexplored area to map out the area more efficiently.

3.3.3 Maze Algorithms [35]

3.3.3.a Wall Following Algorithm

This is a simple to implement algorithm where the robot traces one side of the wall. This can be implemented by using the LiDAR sensor on the robot. With LiDAR detecting the distance between the robot and the wall, the robot can keep a distance to the side of the wall. When the wall is no longer detected the robot will turn in the direction of the side it was tracking. On the other hand, when a dead end is detected, the robot will turn opposite to the direction it was tracking

3.3.3.b Depth First Search

When the LiDAR detects an empty area on at least 3 of its 4 sides it is likely that the robot has approached a junction. The robot will then choose a path to explore. If no heated tin or NFC tag were found, the robot will return to the previous junction and explore the other path.

3.3.3.c A Algorithm*

When the thermal camera detects the heated tin and is at the ball loading area. An A* algorithm can be used to determine the shortest path between the loading area and the heated thin can. By determining each 35cm by 35cm square (sum of diameter of LiDAR and minimum range of LiDAR) as a node, the

algorithm calculates the value of ‘f’ which is the sum of ‘g’ (distance between the loading area and the node) and ‘h’ (the distance between the node to the heated tin). The path determined would be the path with the smallest sum of ‘f’ value.

3.3.4 Program Design

The program design has been elaborated in detail in Appendix B.

3.5 Mode of Operation

3.5.1 Testing and Calibration Mode

During this mode, various scripts can be run to calibrate the sensor by collecting real world data. The values collected can then be used in the autonomous operation. Additionally, during the test run of the robot, the programmable LED and audio beep sequences can be used to indicate the status of different processes to allow for quick troubleshooting.

3.5.2 Autonomous Operation

The robot will operate autonomously receiving inputs only from the sensor.

Stage 4: BOGAT

In this step, analysing the different proposed options, a cohesive solution tailored to the team's problem statement will be developed.

4.1 Analysis of Launching mechanisms

	Accuracy	**Complexity**	**Integration**
Flywheel	**High:** High accuracy	**High:** Requires a geared system	**Hard:** Integration with electrical and software for this is simple as not many additional components are required. However, the integration with the storage system is tricky as the bulky design of the flywheel means less space for storage.
Solenoid	**Low:** Firing plate is used so that force is applied evenly	**Low:** It requires only wire coils	**Medium:** Solenoid will be firing the ping pong ball forward with a signal to the switch hence it does not require any complicated programming. As for the electrical system, a boost converter may be needed in the event that the solenoid is not strong enough. The solenoid will be compact and balls can easily be fed in front of it.
Catapult Arm	**High:** Difficult to aim at a location	**Low:** Consists of a catapult arm and a simple firing mechanism	**Hard:** The program will be required to determine the distance from the robot to the target to relate to the power needed to launch the catapult. The catapult is a large mechanism and the loading mechanism will be complicated due to the leeway required for the large movement of the catapult arm.
Cam Based Shooting Mechanism	**High:** Barrel provides good accuracy	**High:** A lot of interchanging mechanical components	**Medium**: The program will be required to determine the distance and angle required for the mechanism to fire. Its bulky design means that there is less space to the storage system and might be hard to attach onto the robot.

Conclusion

Due to its low complexity, simple integration with software and the storage system, the solenoid has been chosen. The advantages of an easy integration outweighs additional technical difficulty in the event that a boost converter is required to increase the firing strength.

4.2 Analysis of Storage System

	Complexity	**Integration with other Systems**
Vertical	Simple to build	It may block the 360-degree LiDAR Sensor no matter where on the robot the column is placed due to the height required to fit 5 balls. (Appendix A Figure 7)
Carousel	Requires moving components	Requires software to control the movement of a carousel

Conclusion

The vertical column structure has been selected as the foundation of the storage system due to its simplicity. To overcome the identified issues with this option, the structure is altered to be a spiralling column around the robot instead. This allows for an increased storage capacity to store 5 ping pong balls without obstructing the LDS and decreases its vertical height. A servo will be incorporated to load the ball from the storage to the firing system.

4.3 Analysis of Maze Solving Algorithm

	Complexity	**Efficiency**	**Edge Cases**
Wall Following Algorithm	**Low:** Easy to implement.	**Low:** Will not be able to directly travel to unexplored regions	Certain mazes cannot be solved using wall following alone
Depth First Search	**Low:** Easy to implement.	**Mid:** Backtracking results in a larger distance covered by the robot	NIL
A* Algorithm	**Hard:** Requires the breakdown of the map and calculation of individual nodes	**Mid:** Ensure that the path with the least cost is taken. However, it requires a map of the maze.	Requires maze to be fully mapped.

Conclusion

The team will be incorporating multiple maze-solving algorithms to utilise their various advantages and functions. As the maze has no dead end, a Depth First Search algorithm can be used to find the location of the heated tin can and the loading zone. The A* algorithm can then be used to determine the shortest path between the loading zone and the heated can.

Stage 5: Preliminary Design

5.1 Overview

Based on the discussions, the team decided on the systems required to achieve the problem statement. Figure 1 below provides an overview of the entire robot. It shows the connections between different components and their integration.

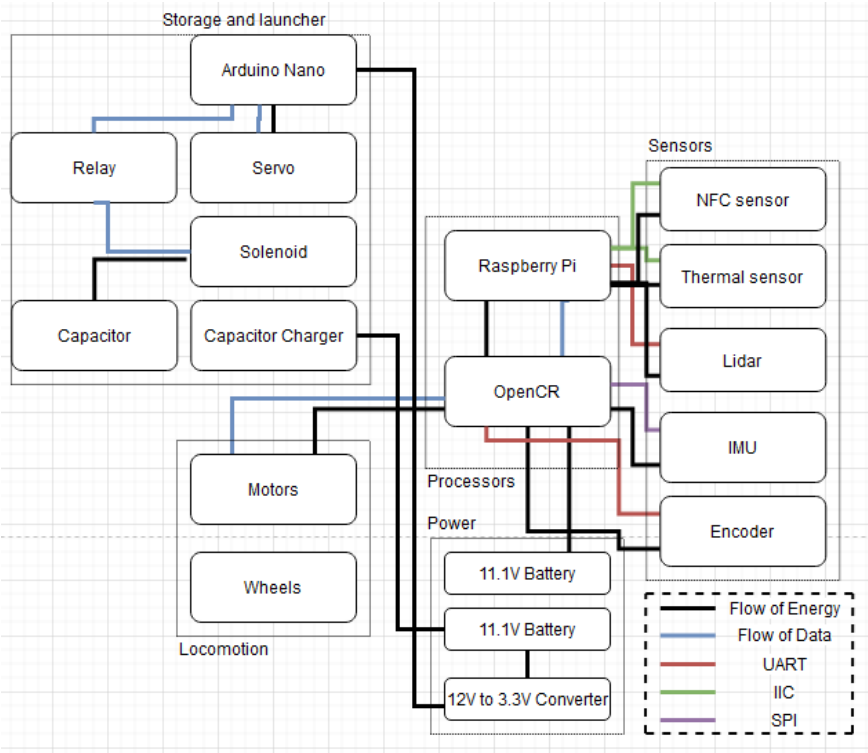


Figure 5: System block diagram

The robot's Raspberry Pi works with the OpenCR to interface with the sensors and motors and provide the computational capabilities to satisfy the problem statement. It will be outfitted with a system of sensors allowing us to map the environment, perform localisation and identify landmarks. The storage system is well integrated with the firing mechanism to strike the targets accurately. The battery provides enough capacity for debugging and the actual test run.

5.2 Software Subsystem

This subsection includes the conceptualised flow of program commands to satisfy the problem statements (Appendix B). Through the flow chart, the team explicitly shows the transfer of data between different systems and the decisions made based on these data. The program contains two strategies, an exploratory program, and a navigating program. For the former, the robot uses the breadth first search algorithm to rapidly explore unexplored regions of the map. The latter utilises the A* algorithm to calculate the shortest path to a specified final destination. Furthermore, to facilitate debugging and calibration operations, the team will be utilising programmable LED and the buzzer onboard the OpenCR as indicators.

5.2.1 Navigation Strategy (revised)

The robot contains 2 modes of operation - exploration and retracing. In the exploration mode, the robot aims to rapidly explore the map by moving towards accessible but unmapped regions. In this mode, it records waypoints when there is a large increase in angular momentum (dead ends). If the tin can is encountered but the balls are yet to be collected (i.e. NFC tags not detected), the current position and the orientation is recorded as landmarks. The retracing strategy is based on the landmarks and takes the shortest path back to its position in order to complete the mission. However, if the whole map has been mapped, indicated through a closed environment with no grey areas (and the tin can or NFC tags have not been found), the robot will return to the recorded waypoints and continue searching for the NFC tag or tin can. Since the NFC tags will be placed along the centre of the path, the robot will be directed to travel along the middle, i.e. the midpoint between walls.

5.2.2 Human Interface (revised)

A LED will be used to indicate if a button press is detected. When the TA presses the button to indicate that the ball loading process has completed, the LED will light up as feedback. Hence, there will not be any accidental case where the TA mistakenly thought that he has pressed the button.

5.2.3 Tin Can Identification and positioning

With the use of a heat sensor array AMG8833, the team will be able to capture an 8*8 grid of the environment ahead of the robot. Using this 2D array, the program will position the heat signature at the centre of the robot while ensuring that the robot's front is facing perpendicular to the wall. This will allow us to position the heat signature at the centre of the sensor by moving the robot back and forth in a linear motion hence reducing the complexity of the task.

5.3 Mechanical Subsystem

Under the mechanical sub-section, the team has decided on the structure of the robot to be similar to the original Turtlebot burger design. Additional components such as the circuit board, storage system and firing mechanism will utilise the existing holes on the turtlebot plate as mounting points. Acrylic plates will be used to protect parts that are exposed and fragile like the circuit board. As concluded from the team's discussion, the team has decided to adopt a spiralling vertical column as the storage system and a solenoid as the firing mechanism. The team will be able to construct a robot that is simple yet effective at satisfying the problem statements (Appendix C).

5.3.1 Ball Trajectory calculation

The robot has to fire the ball at least 50cm away from the tin can that is approximately 30cm tall. Since the solenoid has to be placed at the end of the feeding tube, it will be approximately 5cm above the ground. To accommodate for leeway, the team would perform the team's calculations with the ball being fired 55cm away. Calculations (Appendix D) determined that the ball will require an initial velocity of about 5m/s.

5.4 Electrical Subsystem

In this section, the team has selected an array of sensors deemed essential to the operation of the robot: LDS, IMU, encoder, thermal sensor and the NFC reader. The LDS, IMU and encoder provide essential data for the SLAM algorithms. The thermal sensor identifies the heat signature so the robot can aim and fire the ping pong balls. A relay will be required to control the solenoid such that the circuit is closed for a split second for force to be exerted on the ping pong ball. An NFC sensor will be attached on the first layer of the robot due to its limited sensing distance. Below are additional parts the team have shortlisted for the prototype design of the robot:

1. Thermal sensor: AMG8833
    a. It is 5V compatible
    b. Communicating via I2C
2. Servo: SG90
    a. This servo can be controlled using pulse width modulation (PWM)
    b. It is operated at 5V
3. Solenoid: 24 V small solenoid
    a. It is operated at 12V
    b. A relay placed in between the battery and the solenoid
    c. GPIO on the Raspberry Pi can be used control the switch
4. NFC reader: PN532
    a. As it is operating at 3.3V a level shifter is required
    b. Communicating via I2C

These components will be connected to the Raspberry Pi that determines the robot's course of action. Hence, by directly interfacing with the Raspberry Pi, redundant processes can be minimised.

5.4.1 Actuation System

To protect the OpenCR and Raspberry Pi the actuation system will be on a separate system, to prevent backflow and short circuits from damaging the component. The system will be powered by a separate battery. A Boost Converter connected to the power source will step up the voltage. Furthermore, the team will be using a capacitor to store sufficient charge to be released to the capacitor. A 24V solenoid will be used in the system.

A servo will regulate the funnelling of the ball from the feeding tube into the solenoid launching mechanism. The servo will be connected to two flaps. The first flap controls when the first ball enters the solenoid catchment. Meanwhile, the second flap, perpendicular to the first prevents the second ball from falling out when the first ball falls out. This coordinated motion can accurately dispense balls from the tube.

5.4.2 Solenoid Calculation

The team plan to overvolt a solenoid rated at 24V to achieve a higher launching speed. Calculations (Appendix E) determine the appropriate components required. The solenoid the team intends to use can be referred to here.

Stage 6: Prototyping & Testing

Between stages 5 and 6 of the project, several amendments were made. The team then performed an assessment of how the newly introduced changes might affect our design choices.

Changes/Clarification to Final Mission	**Assessment of changes**
NFC tags placed closed to walls	Navigating near the wall is a more attractive option as the robot can scan for NFC tags.
Close and connected maze with no dead ends	Wall following is less likely to fail
Ping Pong ball has to leave the payload with appreciable velocity, but can be fired from any distance as long as the payload is not in contact with the target.	The robot does not have to fire the ping pong ball from a long distance
Time taken to map the maze is a new judging criterion.	Mapping the maze should be a priority in mission planning

6.1 Mechanical Design

6.1.1 Connection between extension and turtlebot

To ensure that the connection between the launching mechanism and Turtlebot is sturdy, the team allocated mounting points across the whole plate. These mounting points are placed sufficiently such that there is no sagging or flexing at the connection point.

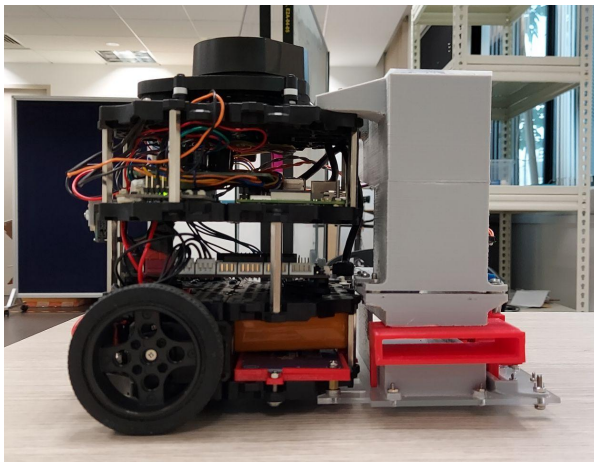


Figure 6.1.a: Side profile of robot

As seen in figure 6.1.a, the base of the robot is completely flat. This proves that the robot has sufficient mounting points to ensure that the extension is well connected to the turtlebot.

6.1.1 Launching mechanism

The launched ball should travel a horizontal distance of at least 30cm. The launcher should also be as compact as possible. Various prototypes were conceived to achieve the aforementioned requirements.

6.1.1.1 Launching Mechanism Iteration 1: Solenoid Launcher

Prototype

The team designed a launching mechanism which utilises a solenoid to impart an impulse on the projectile.

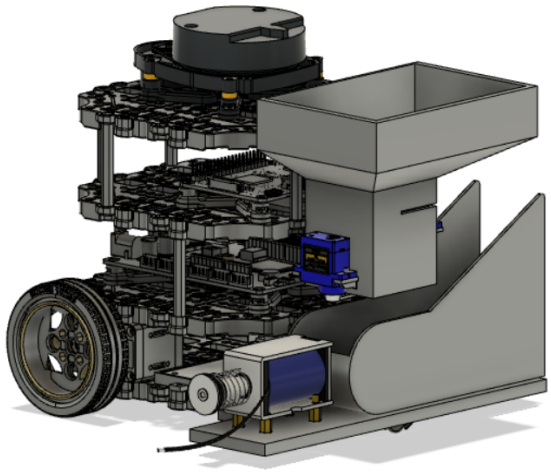


Figure 6.1.b: Solenoid system attached to the back of the turtlebot

In this design, the launching mechanism which consists of the solenoid, inclined ramp and the storage system was attached to the back of the robot. As the front of the robot had two wheels by the side, the team opted to mount the mechanism on the back for additional leeway. The components of the mechanism are modularly mounted onto the robot such that it can be easily removed for updates in design or maintenance.

Testing

The team purchased a solenoid from Sim Lim Square. The solenoid was set-up such that it has an incline of approximately 30 degrees from the table. The team then measured the distance travelled by the ball and found that the ball has insufficient velocity to be launched from the ramp the team designed. The calculations to determine the initial velocity of the ball can be found in Appendix F.

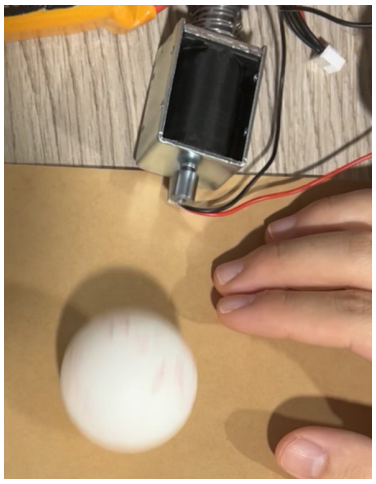


Figure 6.1.c: Testing the launching capability of the solenoid on a set up similar to the digital design

Conclusion

The ping pong ball did not gain sufficient velocity to travel a distance of 30cm when launched. Hence, this prototype launcher is a failure. The team will need to rethink the mechanism used to launch the ball.

6.1.1.2 Launching Mechanism Iteration 2: Flywheel Launcher

Prototype

In the second revision of the launching mechanism, the team opted for a flywheel system. In this system, a flywheel is used to impart the velocity on the ball.

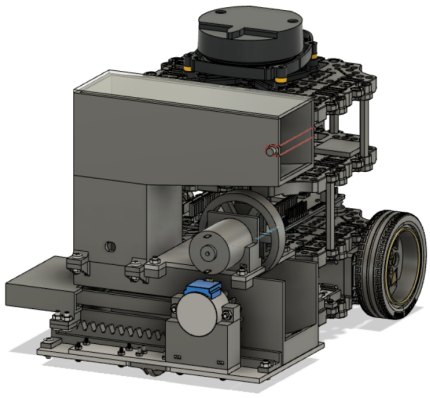


Figure 6.1.d: Flywheel mechanism on the back of the turtlebot

The ramp has a curvature that matches that of the flywheel. This allows the ball to maximise contact with the flywheel so as to maximise the transfer of energy to the ball. The size of the wheel was chosen to match a sturdy plastic wheel that can be found in abundance in the lab's inventory. Hence, the team was able to promptly replace the flywheel in the event that it is damaged under stress.

The team uses a single flywheel design as it requires fewer components and occupies less space compared to a double flywheel design.

Since the team decided not to increase the height of the robot, the robot's launching system was made to be wider. Also, reusing the design from the previous iteration of the launching mechanism, the team decided to mount the mechanism at the back of the robot.

Testing

The team 3D printed the ramp and attached anti-slip mat onto the plastic wheels. After integrating these 2 components, the team then tested the horizontal distance a ball can reach after being launched. At 100% power, the robot launched the ball approximately 4m. The video of the test can be found here.

Conclusion

The flywheel system is an effective mechanism to launch the ping pong ball. The increased power of the flywheel gives greater flexibility in terms of choosing a range to launch the ping pong.

6.1.2 Storage and Feeding Mechanism

The team requires a feeding mechanism that can reliably transfer the ping pong ball from the robot's storage component to the launching component. Based on this requirement, various prototypes were conceived.

6.1.2.1 Storage and Feeding Mechanism Iteration 1: Funnel & Double Servo with Light Gate

Prototype

The team designed a feeding system in conjunction with the solenoid launching system. To feed the ball into a position to be launched by a solenoid, the team utilised 2 servos and a light gate. A funnel-shaped storage will hold up to 5 balls.

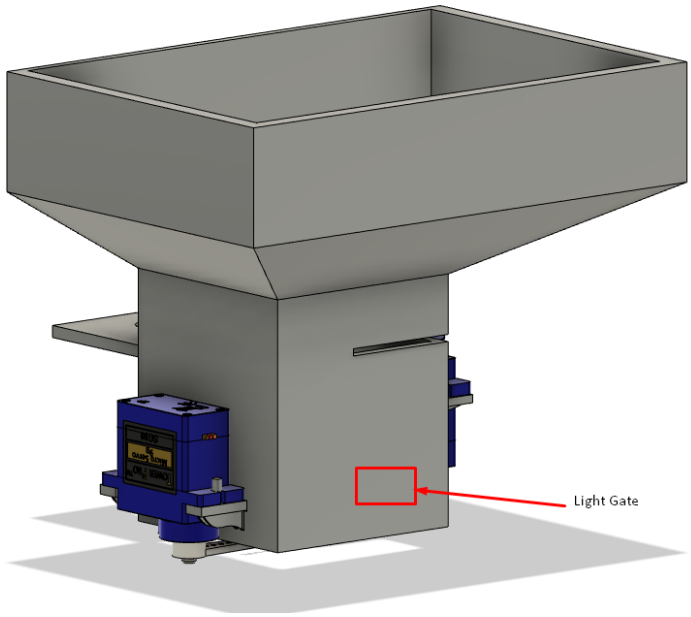


Figure 6.1.e: Funnel with Servos and Light Gate

The bottom servo will swing open to feed the ball into the path of the solenoid. The bottom servo will then swing back to close the gap. The top servo will then open to allow a ping pong ball to fall between the 2 servos. The light gate will check for the presence of a ball before closing the top servo.

Analysis

This system is complex as coordination between several components are required. Furthermore, two separate moving parts increase the possible points of failure.

Furthermore, while trying to integrate the funnel with the rest of the robot, the team realised that the bulky nature of the funnel (figure 6.1.e) caused the robot to be larger than expected. Additionally, the team found that movement from the robot may cause the ball to bounce around, occasionally blocking the LiDAR. Lastly, as the team was considering 3D printing the funnel, the irregular shape of the funnel will require a lot of supporting materials. This means that more resources and time will be required to 3D print the funnel.

Conclusion

This method was deemed ineffective as it is hard to integrate with the rest of the robot. Furthermore, the fabrication of the parts will not be easy. To improve this design, the team envisioned a system that blocks and pushes the ball out in a singular motion.

6.1.2.2 Storage and Feeding Mechanism Iteration 2: L-shaped Tube, Rack, Pinion and Slider

Prototype

The team designed a feeding system using a rack and pinion. This feeds the ball into the flywheel while stopping the rest of the ball from entering the feeding lane. The team adopted this feeding system as it is a compact solution that can reliably feed the ping pong ball into a fast-spinning flywheel. An animation can be found here. Using an L-shaped tube with ping pong balls inserted from the side results allows for a more compact design.

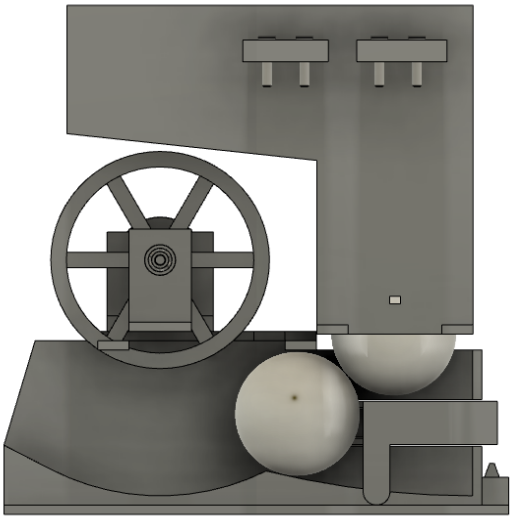


Figure 6.1.f: Cross-sectional view of Rack and Pinion system

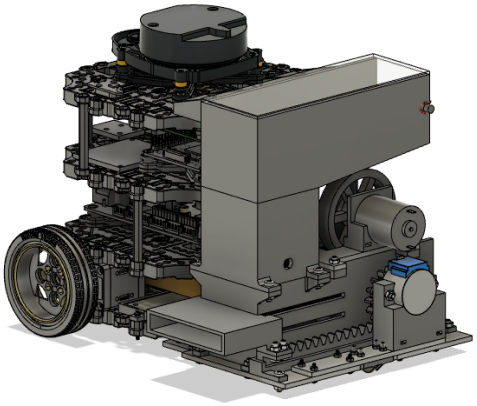


Figure 6.1.g: Side View of Storage and Feeding System

Testing

The storage and feeding system was 3D printed and tested. A video of the test can be found here. Through the test, it was determined that the dual function of feeding and blocking other balls is effective. The back and forth motion of the pinion was much easier to program as well.

The storage system was able to fit all 5 ping pong balls easily. The lid above the storage system also prevents the ping pong ball from bouncing up and affecting the LiDAR readings. The L-shaped tube was also easily printed without much supports required.

Conclusion

Due to the reduction in the number of components and the simplicity of the program, the team decided to implement the rack, pinion and slider feeding system for the robot. The more compact storage system that does not block the LiDAR is preferred.

6.1.3 Summary of Iterations

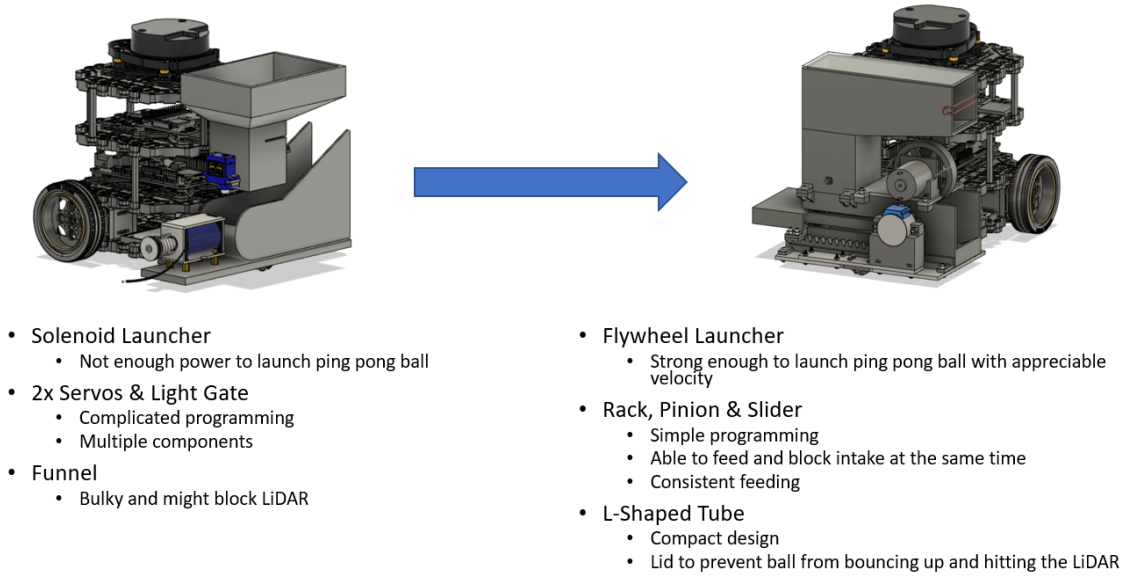


Figure 6.1.h: Summary of Changes and Improvements

6.2 Electrical Design

6.2.1 Electrical Component Testing

Component	Things to Test	Outcome
PN532 Breakout Board	If PN532 can detect NFC at planned position	PN532 is able to detect the presence of NFC tags at planned position
28BYJ-48 Stepper Motor and ULN2003 Driver	Has the ability to move the slider and push a ping pong ball	Stepper motor is strong enough to move the ping pong ball
DC Motor, Motor Driver and Flywheel	Has the ability to propel ping pong balls up to 35cm away	Ping pong balls were propelled more than 4m away. Video
LDS-01 (LiDAR)	Consistency of LiDAR	The team noticed that by bending the wire between the LDS-01 and USB2LDS the LDS-01 was sending invalid values. Sometimes the LDS-01 will send invalid values.
AMG8833	Ability to detect heated tin	AMG8833 is able to detect the heated tin at about 50cm distance
LDR	Raspberry Pi is able to read values from the LDR	Raspberry Pi is able to read values from the LDR
Button on OpenCR	Ability to detect button press	Able to detect button press. However the location of the button is quite inaccessible

6.2.2 LiDAR (LDS-01)

When the wire connected to the USB2LDS device is placed under stress, the values from the LiDAR become corrupted. The team resolved this problem by shifting the location of the USB2LDS device, ensuring no stress is exerted on the wire.

6.2.3 Light Dependent Resistor (LDR)

Raspberry Pi is unable to take in analog inputs. Thus, an analog to digital converter (ADC) would be required to take in values from the LDR. This was undesirable as the ADC is bulky and occupies additional space on the third layer. It is also not compatible with the Raspberry Pi Hardware attached on top (HAT) that the team are planning on making. After researching, the team found that by connecting a

capacitor in the circuit, and with the help of the library the robot was able to take readings from the LDR [39]. The team tested this and obtained satisfactory readings as shown in the table above.

Due to the different requirements of the robot as the team created different prototypes, the LDR was used differently.

6.2.3.1 Iteration 1: Light Gate

The LDR was originally planned to be used in conjunction with an LED to act as a light gate. The light gate was used to determine the presence of ping pong balls at the bottom of the funnel. However, upon testing with the funnel, the team noticed that the ping pong ball never got stuck in the funnel. Hence, with the confidence that the ping pong ball will always fall to the bottom of the funnel, the presence of the light gate is no longer required and the program can be simplified.

6.2.3.2 Iteration 2: Loading Zone Detector

The LED and LDR was repurposed to act as a sensor to detect the loading zone signs (which is of a different colour to the ground). For this purpose, the team conducted a test and determined that the robot could differentiate between plain ground and the loading zones. However, due to the increased reliability in the navigation algorithm of the robot after several iterations, the robot is able to consistently detect the NFC tags in the maze. Hence, having the LDR is redundant.

6.2.3.3 Iteration 3: Loading "Button"

As discovered in prior tests, due to the inaccessibility of the button on the OpenCR, the button is not suitable as a viable option for human interaction. Hence, the LDR was repurposed to act as a replacement "button". By shining a torchlight on the LDR, the robot will be notified that the ping pong balls are loaded. The team thus conducted several tests and determined that the values from the LDR when it is under a torchlight and when it is not can be differentiated with ease. Hence, the team proceeded with repurposing the LDR by shifting its position to the top layer of the Turtlebot.

The team found and resolved issues with the LDS-01 and the OpenCR buttons. Other components produced satisfactory results that do not require further revisions at the moment.

6.2.4 Revisions Summary

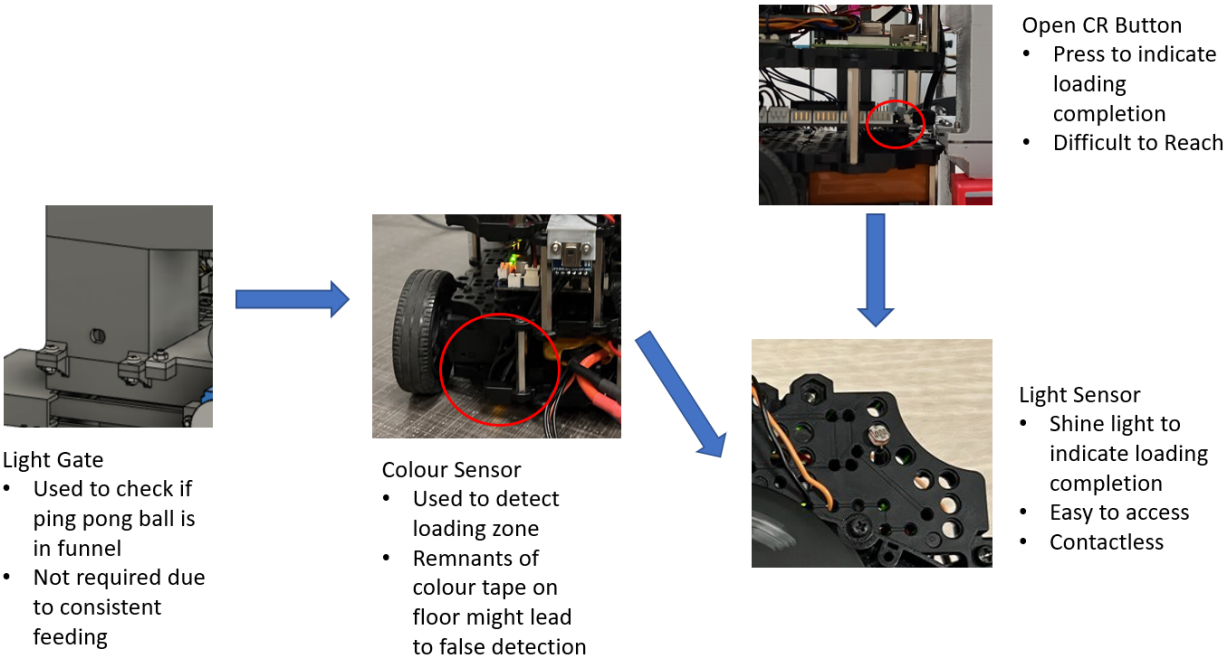


Figure 6.2.a: Summary of Changes and Improvements

6.2.5 Power Consumption

Power Drawing from Raspberry Pi 3B+

Component	**Voltage (V)**	**Current (A)**	**Power (W)**
PN532 Breakout Board	3.3	0.0	0.1
LDS-01	5	0.4	2
AMG8833	3.3	0.004	0.01
LED	3.3	0.002	0.02

Maximum current draw from Raspberry Pi's 3.3V pins is 0.5A. The PN532, LED and AMG8833 are drawing less than the maximum amount.

Power Drawing from Open CR

Component	**Voltage (V)**	**Current (A)**	**Power (W)**
Raspberry Pi 3B+	5	0.968	4
Stepper Motor & Driver	5	0.5	2.5

The total current draw of 1.5A at 5V is below the Open CR's limit of 4A.

Power Draw from Battery

From tests conducted (Appendix G), the maximum current draw from OpenCR 1.0 will be around 1A, while the DC motor will have a maximum current draw of 4A. Therefore, the total maximum current draw of 5A is way below the battery's maximum current output of 63A.

Additionally, the tests concluded that the battery can support an estimated run time of 90 minutes, which is greater than the maximum time of 30 minutes.

6.3 Software Design

6.3.1 Navigation

Navigation is essential for the robot to find its way around the maze and for mapping the environment. It is also essential for the robot to locate the different elements of the maze such as the NFC tags and the heated object. To develop a robust navigation algorithm that can achieve the team's aforementioned objectives, several prototypes were conceived.

6.3.1.1 Navigation Iteration 1: Frontier Detection and waypoint searching

Prototype

By default, the Cartographer's SLAM algorithm generates a map with a resolution of 5 cm and updates the map at 1 Hz. However, the resolution of 5 cm is too large for the team's application of processing the map and the low update frequency means that the responsiveness of the algorithm to changes in the environment is low. Hence, by changing the configuration files of the Cartographer's SLAM algorithm, the team can obtain a map with higher resolution and higher update frequency.

Using image manipulation techniques, the program derives the boundaries of unmapped regions. With the coordinates of these regions and the coordinates of the robot relative to a global map, the program then charts a path from the robot's present location to the unexplored region using the A* algorithm. Of the different types of path searching algorithms, A* has the fastest computation time. Optimizations were performed on the algorithm for it to achieve even faster computation. The A* algorithm also considers the size of the robot so that the robot will not run into any obstacles. Hence, this rapidly pushes the frontiers on the map and explores the entire maze in the shortest possible time.

The team's A* algorithm is designed to generate a path that minimises turning. Additionally, other algorithms are used to smooth the charted path so that the movement of the robot is smooth (does not involve consecutive turns). This algorithm processes the path such that the program can send short movement commands that the robot can easily follow. The algorithm sends two types of commands, turning a specific angle and moving forward a specific distance.

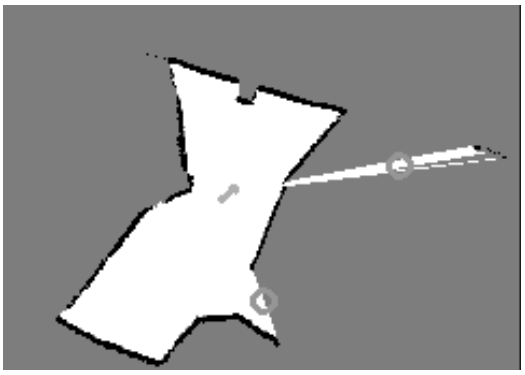


Figure 6.3.a: Frontier detection algorithm output

The team verified the frontiers detected are indeed frontiers of the explored region. As seen in the Figure 6.3.a, grey circles are drawn on the location of the different frontiers and the arrow represents the location of the robot and its heading. Hence, the algorithm can accurately identify the boundary of the unmapped region. A video showing the algorithm updating the frontier in realtime can be found here.

Figure 6.3.b: Path to the nearest frontier generated using the A* algorithm

With the coordinates of the unmapped boundary, a path searching algorithm is used to find the path from the current robot position to the target location. As seen in Figure 6.3.b, a black line from the arrow to the circle represents the path the robot should take to travel to that frontier. Hence, this shows that the A* algorithm is able to find the shortest path that minimises turns.

However, the time taken to compute a path in a 160 by 175 grid takes approximately 0.4 seconds. If the point does not have a valid path to it, it takes up to 2 seconds to compute that there is no possible path. Given that there are many frontier points and some cannot be reached, the program takes up to 10 seconds to compute a viable path.

After obtaining the path from the A* algorithm, the team ran the path smoothing and processing algorithm and sent the movement instructions to the robot. While the robot was able to execute the turns precisely, it had difficulty moving the required distance accurately.

Conclusion

In conclusion, while both the frontier detection algorithm and A* algorithm worked individually, when integrating both of them together, the team realised that the computation time for a path using A* was too slow for the team's use case. Furthermore, the inability of the robot to move a specified distance accurately suggests that it cannot follow the charted path slowly, hence making it unreliable.

6.3.1.2 Navigation Iteration 2: Simple Wall Following

Prototype

In the second revision of the team's navigation algorithm, the team chose an algorithm that is easy and fast to develop. This ensures that the robot has a reliable navigation alternative in the case the team's more ambitious endeavours fail. This wall following algorithm navigates using a decision tree based on the nearest obstacles from the robot in 3 selected directions. As seen in Figure 6.3.c, the simple wall following algorithm is based on a decision tree that the team constructed.

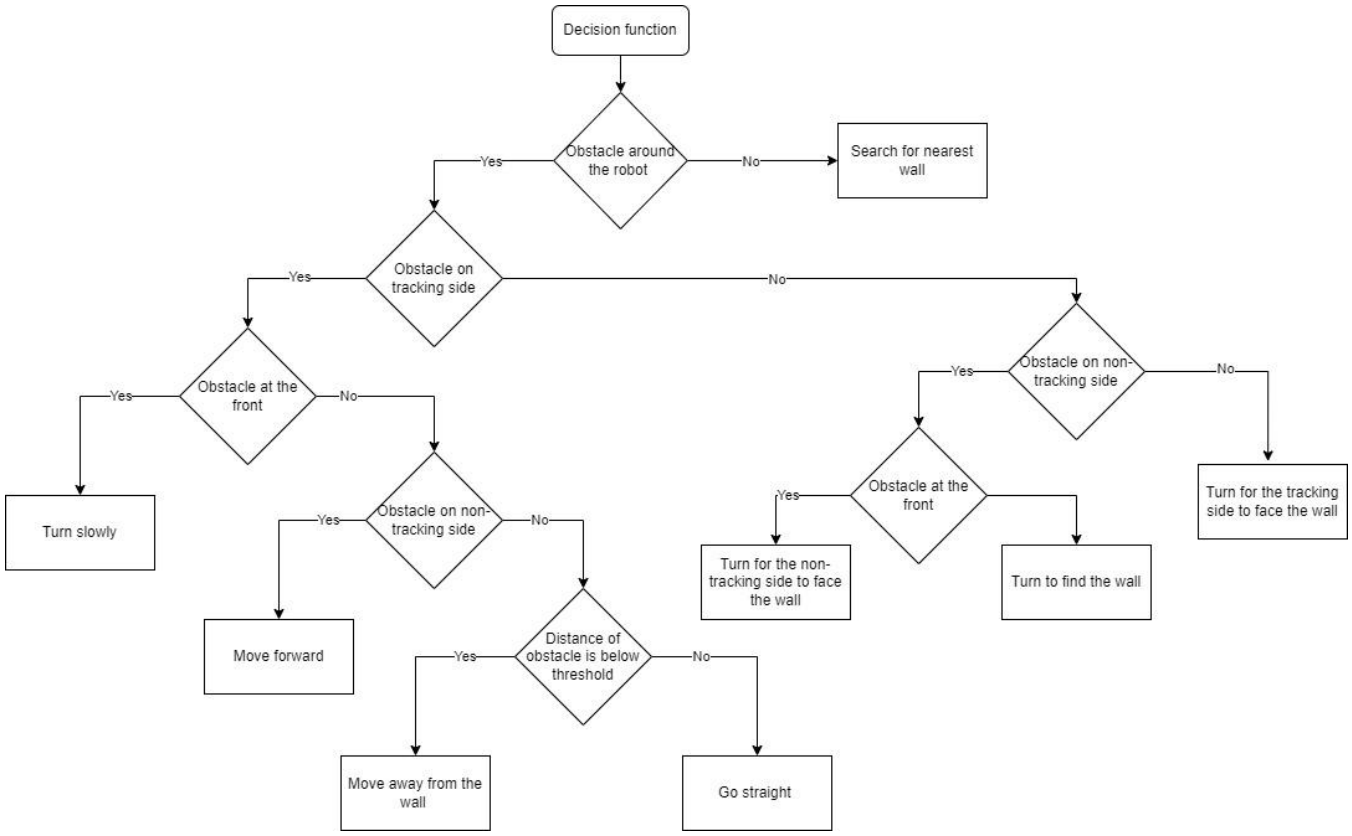


Figure 6.3.c: Decision Tree for Simple Wall Following

Testing

After completing the decision tree, the team then tested the performance of the robot on a maze. The team fine-tuned parameters in the program to achieve smooth movement. Furthermore, the team also placed the robot in extreme positions, to stress test the algorithm.

Despite adjusting the parameters, it was found that the robot moved in a sinusoidal wave pattern next to the wall. Furthermore, when placed in extreme positions, the robot failed to correct its position without colliding with the wall. Occasionally, it took too long for the robot to correct its position.

Conclusion

Due to the sinusoidal movement of the robot, the robot requires a larger leeway in terms of its distance from the wall to ensure the robot does not collide with the wall in any situation. However, given that the NFC tags were placed close to the wall, approximately 20cm away from the wall, the robot struggled to read the NFC tags.

To make the robot's movement smoother, the team took inspiration from the PID control method [37] and proportionally changed the angular velocity of the robot based on its distance from the wall. After assessing the feasibility of this algorithm, the team realised that it cannot be implemented without using another point of reference along with the one the program already uses. This gave the team the idea of the triangle wall following.

6.3.1.3 Navigation Iteration 3: Triangle Wall Following

Prototype

The team came across a paper describing an implementation [38] of the concept of triangle wall following which ensures the robot can maintain a bearing parallel to the wall it is tracking. The paper described a series of calculations performed to measure the bearing of the robot from the wall (Appendix H). With reference to Figure 6.3.d, the team can apply a proportional control on the bearing of the robot with respect to the wall, θ (which can also be viewed as the angle of correction of the robot). This allows the robot to rapidly correct its direction of movement and follow the wall in a smooth manner.

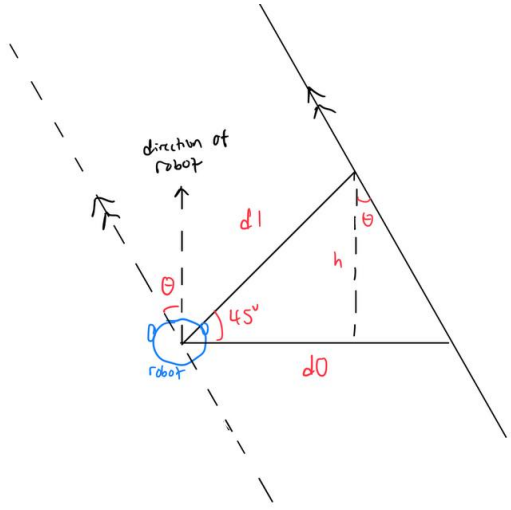


Figure 6.3.d: Calculating the bearing of the robot with respect to the wall

Triangle wall following can only ensure that the robot can move in a direction that is parallel to the wall. It does not ensure that the robot can maintain a specified distance from the wall. Hence, the team devised a method to integrate the distance into the proportional controller (Appendix H).

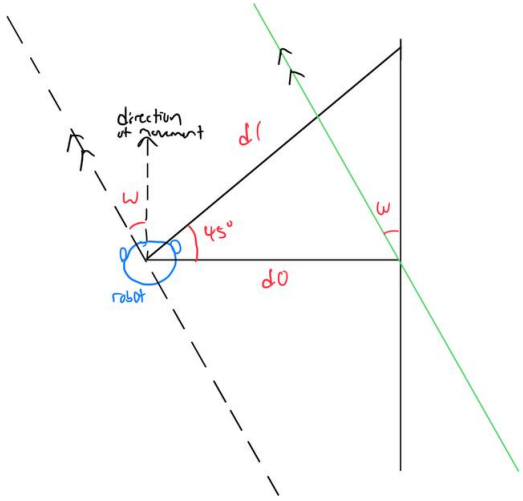


Figure 6.3.e: Augmenting the wall based on the distance of the robot from the wall

Based on the distance of the robot from the wall, the program can augment the bearing of the wall it has to track. With this concept, the team utilises another proportional controller to vary the angle of the wall. This improved algorithm can factor in both the robot's present bearing and its distance from the wall. Based on figure 6.3.e, if the distance between the wall and robot, d_0, is smaller than the target distance, ω will be the angle of correction for the robot for it to move away from the real wall (black) by tracking the augmented wall (green).

Testing

After implementing the new algorithm based on the team's calculations, the team fine-tuned the parameters of the proportional controller and then placed the robot in a maze to test the algorithm. The team found that the robot is able to accurately maintain the specified distance from the wall and can follow the wall smoothly. Furthermore, when placed in extreme positions, for example when the robot is far from the wall or the direction of the movement is not parallel to the wall, the algorithm allows the robot to return back to the specified distance swiftly.

Conclusion

The triangle wall following algorithm is effective in maintaining the heading of the robot and distance of the robot from the wall. Furthermore, it is responsive to any changes in the contours of the wall.

6.3.1.4 Navigation Iteration 4: Scanning a range of distance

Prototype

The team realised that due to the imperfections of the maze, such as gaps between cardboard and certain parts of the wall may have protrusions, the program required a way to find the actual shortest distance from any obstacles around the robot. This means that instead of using only a single distance point to measure obstacles in front or at the side of the robot, the program can find the minimum value of distances from a range of angles from the robot.

It was found that this new addition to the algorithm will lead to the detection of false positives of obstacles if the direction of movement of the robot is not parallel to the wall. In that case, a solution was formulated which allows all of the robot's distance measurements to be more accurate.

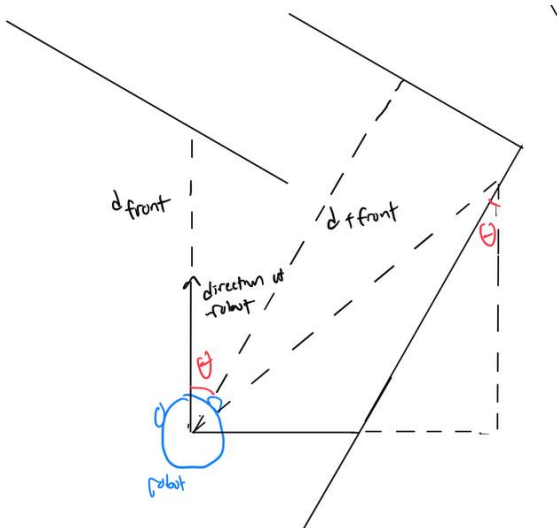


Figure 6.3.f: Measuring the true front distance of the robot

From figure 6.3.f, d_{tfront} represents the actual distance of the obstacle in front of the robot while d_{front} represents the distance of the obstacle in the direction the robot is facing. If the d_{front} is used, this would lead to false positives in the detection of obstacles in the path of the robot, causing the robot to execute a turn instead of moving forward. Hence, by solving for θ in Figure 6.3.f (Appendix H), the programme is able to find d_{tfront} and therefore obtain more accurate measurements of the obstacles around the robot. The similar concept can be applied not only to the front of the robot but also to the sides.

Testing

The team intentionally allowed gaps in the walls of a maze we constructed. In this poorly constructed maze, the team found that the robot is able to navigate it smoothly when a range of distances is used instead of only one specific distance to find the distance of the obstacle.

When the team set up a maze that resembles that in Figure 6.3.f, as expected, the robot failed to follow the wall as it detected the other perpendicular wall prematurely. When the scanning range is corrected based on θ, the robot was able to smoothly navigate through the maze as intended.

Conclusion

By using a range of distance information from the LiDAR that is calculated based on the robot's direction of movement versus the wall it is tracking, the team is able to increase the reliability and accuracy of the robot's measurements of the distances to the obstacles.

6.3.2 Summary of Navigation Iterations

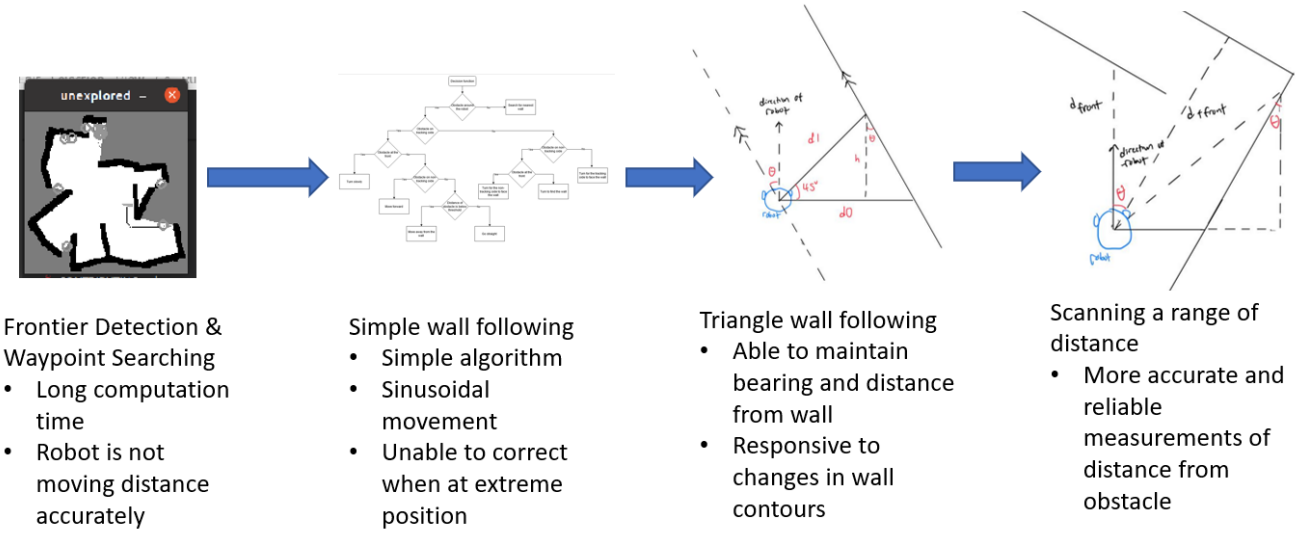


Figure 6.3.g: Summary of Changes and Improvements

6.3.3 Shooting sequence

After detecting the heated object, the robot needs to navigate itself into the range of the heated object. This robot has to ensure it is approximately 30 cm away from the tin can. Furthermore, sufficient turning space is required to ensure that the robot does not collide with any obstacles while attempting to align its launching mechanism with the tin can. With the aforementioned requirements in mind, several prototypes were conceived.

6.3.3.1 Shooting sequence Iteration 1: Path searching

Prototype

The team repurposed the code from the team's frontier detection algorithm from 6.3.1.1 to instead navigate the robot towards the heated target. After the thermal sensor identifies the heated object, the corresponding location of the object is marked on the tin can. As such the starting point and the ending point is passed into the A* path searching algorithm, allowing the robot to navigate to a position 30cm away from the tin can.

Once the robot is in position, it will then aim the launching mechanism towards the heated object. This

will be done by calculating the amount to turn based on the information from the map.

Testing
The team constructed a maze with a heated object and intentionally placed the robot in close range of the heated object. However, just like finding a path to the frontier, the robot was unable to follow the planned movements accurately. The robot constantly ended up at end points that were not as intended by the path generated. Furthermore, the long duration that it takes to generate a path means that the path cannot update while the robot is moving. Thus, the program is unable to chart a new path for the robot the moment it deviates from the path.

All of these cause the robot to repeatedly try to work its way to a location that fulfils the condition only for the robot to launch the ball without success.

Conclusion
Using A* to search for a path to the heated object is not suitable due to the imprecise movements of the robot.

6.3.3.2 Shooting sequence Iteration 2: Simple obstacle avoidance

Prototype
Once a heat signature is identified, the robot will move straight towards it. While doing so, the robot will constantly ensure it faces the centre of the heated object. If there are any obstacles detected in close proximity to the robot, it will move away from the obstacle. The robot will repeat these two simple steps until it is in position to launch the balls.

Once in position, the robot will execute a specific turn that will align the launching mechanism towards the heated object.

Testing
The team constructed a maze with a heated object and intentionally placed the robot in close range of the heated object. While the robot took a long time to move towards the target, it consistently hit the target. The team placed the robot in more challenging mazes where the walls leading to the heated target were narrower or the heated target was placed in tough positions. The robot was able to find its way to the heated target while ensuring that it had sufficient space to rotate and align the launcher towards the heated target.

Conclusion
This simple yet effective method of moving the robot towards the heated target is able to produce consistent results.

6.3.4 Summary of Firing Iterations

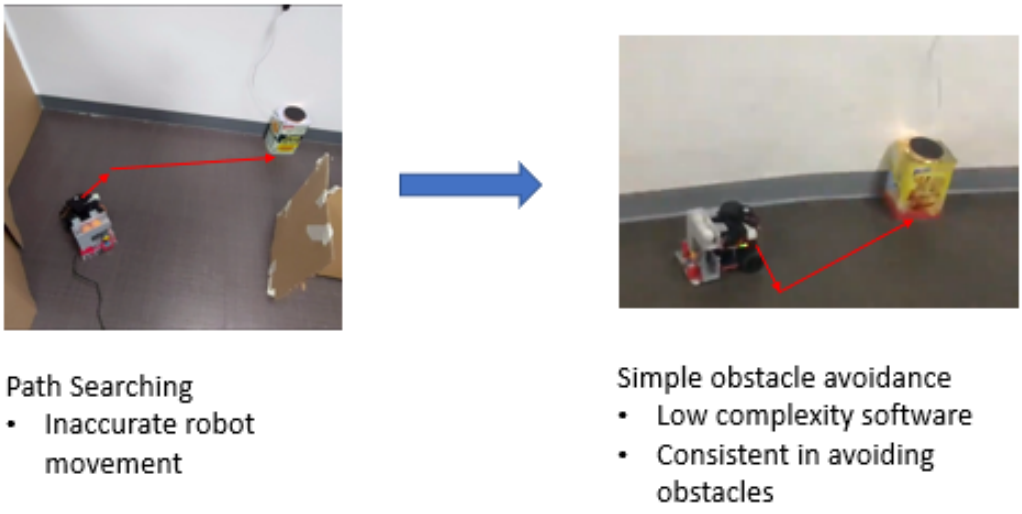


Figure 6.3.h: Summary of Changes and Improvements

6.3.5 Raspberry Pi control

On the Raspberry Pi, the team wrote a program to read from the sensors on the robot, namely, the NFC reader, thermal sensor and LDR. The Raspberry Pi reads the LDR, thermal sensor and NFC reader at 10Hz.

Spikes in the LDR values arising from electrical noise may affect the consistency of the program. Hence, a moving average was used to smooth the data from the LDR. This ensures a torchlight has to be shone for a short period of time before the program acknowledges the signal. The moving average prevents accidental detection.

For the flywheel launcher, a firing sequence is preprogrammed on the Raspberry Pi.

Stage 7: Critical Design

After concluding on the prototyping process, the team further refined each component by making minor tweaks. With these refinements, each component can perform its tasks reliably. In this stage, the team will describe how the design fulfils the problem statement and the rationale behind the team's design choices.

Images of the completed robot can be found in Appendix I.
Labelled CAD of the team's robot can be found in Appendix J.
General system assembly can be found in the End User Document
Detailed Assembly Document can be found on Github.

7.1 Mechanical Design

7.1.1 Ball Storage Subsystem

To prevent the ping pong ball from bouncing when the robot is moving and blocking the LiDAR, a lid is placed on top of the funnel to restrict the ball's vertical movement. Additionally, the funnel is designed to store 5 ping pong balls to maximise the chances of the ball hitting the target. The funnel design allows for the ball to be fed passively into the slider.

7.1.2 Launching Subsystem

The launching system was designed to be compact and with minimal number of components. Hence the team proceeded with a single flywheel design. Additionally, the use of a stepper motor ensures consistency in feeding the ping pong ball into the flywheel. This reduces the chance of the ball being trapped in the subsystem. Calculations (Appendix K) were done to ensure that the components were sized sufficiently to launch the ball at the target with appreciable velocity.

7.1.3 Modularity

The payload was designed to be modular. By removing 8 screws, the launching and ball storage system can be removed from the turtlebot. This allows for easy maintenance of parts that might be damaged due to wear and tear. Additionally, it allows for the mechanism to be easily transferred to other Turtlebot Burgers and for the Turtlebot to be attached to other elements, allowing for greater versatility.

7.1.4 Centre of Gravity

The system was designed to keep the centre of gravity low. This has been accomplished by not adding the payload in between the Turtlebots layers and locating the heavier components (e.g.: DC motor, stepper motor and battery) near the base of the robot. The low centre of gravity results in better stability when the robot is moving.

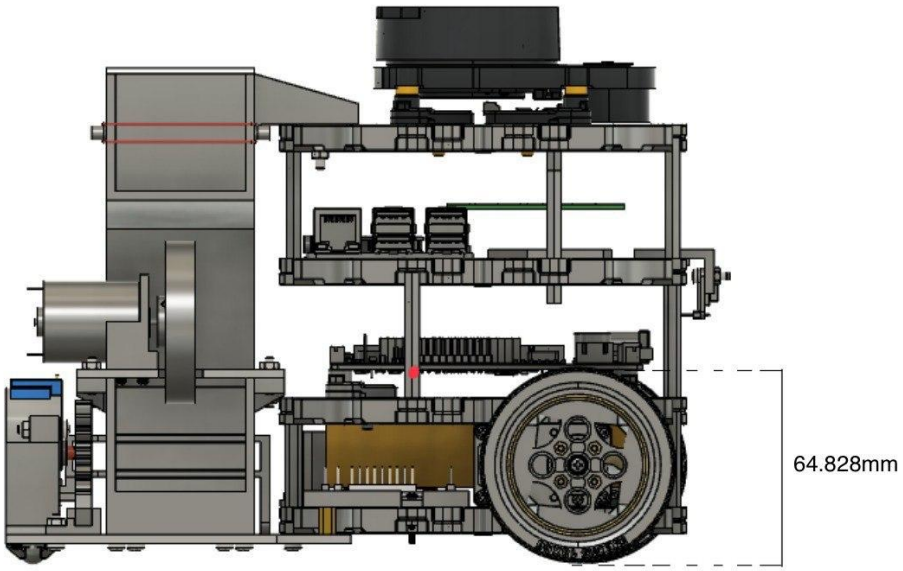


Figure 7.1.a: Side View of Robot and its Centre of Gravity represented by the red dot

7.1.5 Sturdiness

The launching and ball storage subsystems are mounted together. The payload as a whole is mounted from the top and bottom of the Turtlebot. Multiple mounting points are used to ensure that no excess stress is exerted on any mounting points. Mounting points on the subsystems are also supported with ribs to increase its ability to support vertical load.

7.1.6 Fabrications

The components were designed to be 3D printed or laser cut. This allowed for rapid prototyping as the team did not have to machine the components individually. Also, since 3D printing prints the part as a whole, the team simplified the assembly process to locking the components together using screws. The low lead time also allows the team to go from designing to testing in a shorter time frame. Additionally, by not utilising more Turtlebot components, it allows us to keep the cost down as an extra Turtlebot set does not have to be purchased.

The rack and pinion was designed to be 3D printed as the team was unable to source a rack and pinion of the required dimensions for the team's design. The team was also unable to find a rack with a compatible mounting point as the slider.

STL and DXF files that the team used to fabricate the robot's components can be found here.

7.2 Electrical Design

7.2.1 Electronic System Architecture

The Electronic System Architecture shown below is a summary of the flow of energy and data throughout the system.

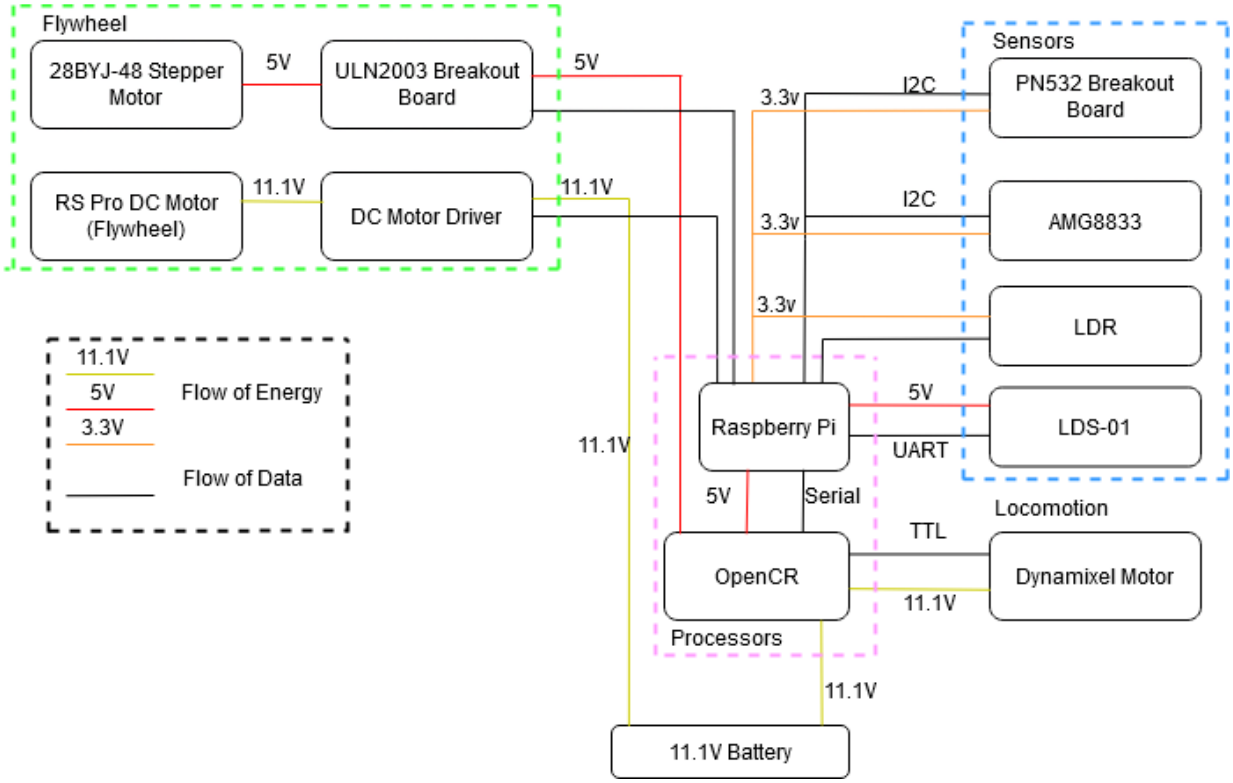


Figure 7.2.a: Electronic System Architecture

7.2.2 Perfboard

The team designed a Raspberry Pi Hardware Attached on Top (HAT) and its schematic can be found here.

	Pros	Cons
Breadboard	- Easily modifiable	- Large number of wires required - Insecure connections
Perfboard	- Easily modifiable - Fewer wires required compared to breadboard - Secure connections	- Have to solder wires and connections manually - Occupy larger space due to the wires
Printed Circuit Board (PCB)	- Smaller than perfboard - Fewer wires required - Secure connections	- Long lead time - Higher cost - Difficult to modify/repair

After much consideration, the team decided to use a perfboard to connect the components together. This is due to the ability to modify the connections easily when needed. Additionally, as the team did not have to wait for the PCB to arrive from overseas, it provided the team with more time for evaluation and testing.

7.2.3 Power

As shown in the electronic system architecture, 11.1 V has to be supplied to the DC motor driver and OpenCR. While 11.1 V can be drawn from OpenCR, it is not advisable to power the DC motor driver and motors from OpenCR. This is as high current draw can damage the OpenCR. Thus, to supply 11.1V to both OpenCR and the DC motor driver, the team manufactured a new cable to split the power output from the battery.

7.2.4 Wiring

The team measured and cut wires to a specific length to prevent wires from extending out of the robots. This ensures that the wires of the robot will not be hooked onto any obstacles when the robot is traversing in the maze and creates a neater look overall without protruding wires.

The wires are colour coded to prevent wrong connections. It also allows for easier troubleshooting and verification.

7.3 Software Design

The repository containing the team's code can be found here.

7.3.1 Mission Planning

As time taken to fully map the maze is part of the grading criteria, the priority of the robot will be to map the maze fully first. To prevent the robot from stopping unnecessarily and taking a longer time to map the maze, NFC tags and heated tins are ignored until the maze is fully mapped. Only after the maze has been fully mapped will the robot stop when an NFC tag is detected.

The software block diagram describing the logical flow of the program can be found here.

7.3.2 Sensors and launcher control

For the NFC reader, the program publishes a boolean indicating the presence of a NFC tag at 10 Hz. For the thermal sensor, a 8 by 8 float array is published. For the LDR, the analog value from the LDR is published. For both the thermal sensor and the LDR, floats are published instead of a simple boolean to ensure that the calibration can be done locally on the laptop. To activate the launcher, an integer is sent from the control laptop to the RPi. This integer represents the speed of the flywheel.

This means that the code on the Raspberry Pi does not need to be modified during calibration.

7.3.3 Wall Tracking

Based on the prototype triangle wall tracking, the team further optimised the parameters for speed. Due to the more advanced triangle wall tracking algorithm, only two variables are required to be calibrated for it to track the wall smoothly. The two variables are the proportional constant for angle of deviation and proportional constant for tracking distance error.

However, there are certain cases when this wall tracking fails. After testing the algorithm to identify all of its points of failures, the team then made special manoeuvres for each condition. Among these conditions are when the robot encounters a 90 degrees turn and when the robot is required to perform a u-turn to go around the wall.

To further improve the reliability of the team's robot, as mentioned in the team's prototype, the team will programmatically determine the actual front and side of the robot no matter what direction the robot is facing. With this the program can more accurately determine the obstacles around the robot and allow the algorithm to decide on the correct manoeuvre.

Furthermore, the team added several fail-safes to prevent the robot from getting stuck in the maze. Firstly, if the robot detects that an obstacle is in close proximity, it will execute a manoeuvre to reposition itself to correct its trajectory. This will allow the robot to prevent itself from crashing into the obstacle. Secondly, if the robot is facing away from the wall, the program will again correct its trajectory. Finally, if the robot detects that its orientation remains constant for a certain time frame despite attempting to move, the robot will determine that it is stuck and hence will move a short distance to detach itself from the wall.

Additionally, the program was written with the flexibility to choose left or right wall tracking. This allows for the robot movement to be optimised based on the maze layout.

7.3.4 Shooting

When the thermal sensor picks a heat signature that has a temperature of higher than 31.5 degrees celsius, the shooting mode will be activated. In the shooting mode, the robot will simply move towards the heat signature until the robot is approximately 20cm from the tin can.

In this shooting mode, the robot will actively ensure that it has sufficient space around it to execute a turn to aim the firing mechanism towards the tin can. However, when the robot detects an obstacle on its side, it will execute a manoeuvre to move away from the obstacle.

When the firing conditions are fulfilled, the robot will rotate to position the launching mechanism to face the tin can and activate the firing mechanism.

This shooting sequence is simple yet reliable in moving towards the target while avoiding any obstacles in the way.

7.5 Bill of Materials

The total cost of the robot comes to $1017.08 while the total number of parts comes to 384. A detailed breakdown can be found here.

The components have been split into 3 categories: Turtlebot, Mechanical and Electrical. Components under the Turtlebot categories have been obtained in the Turtlebot Kit and used to build the initial Turtlebot Burger.

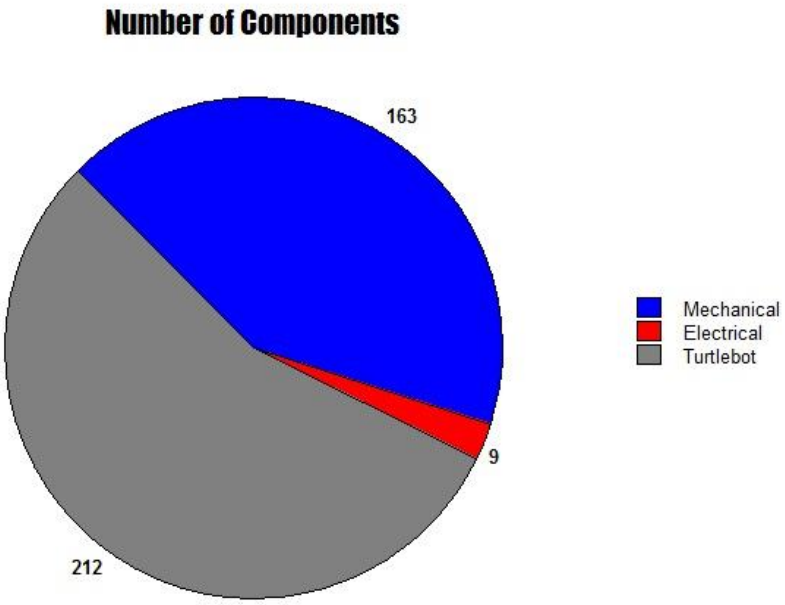


Figure 7.5.a : Breakdown on the Number of Components

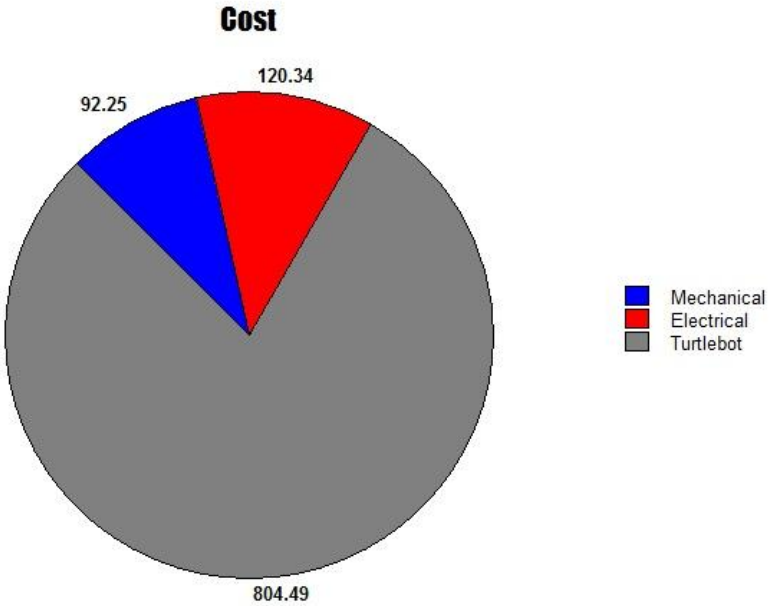


Figure 7.5.b: Breakdown on the Cost of Components

Stage 8: Testing & Evaluation

8.1 Testing

8.1.1 Hardware Integration

8.1.1.1 Orientation of Screws

When the robot is moving, loud screeching sounds can be heard. It was determined that the tip of the screw at the rear of the robot was scratching the floor. To mitigate the issue, the orientation of the screws were flipped.

8.1.1.2 Integration with Perfboard

After placing the perfboard on the Raspberry Pi, the team had trouble attaching the top layer. The team noticed that this is because the Raspberry Pi is elevated in the original Turtlebot Burger build. To remedy the issue, the team mounted the Raspberry Pi directly to the Turtlebot plate instead. The team used screws, nuts and the screwhole on the Raspberry Pi to secure the Raspberry Pi.

8.1.2 Software Integration

8.1.2.1 Large Turning Radius

During the movement of the robot, it was observed that the robot has a large turning radius. To cater for the larger distance required for the robot to turn, several provisions were made in the program.

Provision	**Reasoning**
Turn at a further distance from the wall in front	Prevents the rear from hitting the obstacle in front when turning.
Track the wall at a further distance	Prevents the rear from hitting the wall when turning
Reverse slightly if the wall is too close	Provide more distance at the front of the robot for it to manoeuvre .
Move forward while turning	As the robot is tracking further away from the wall and is turning at a further distance from the wall in front, moving forward and turning allows the robot to be closer to the wall after turning.

8.1.2.2 Invalid values from sensors

The team ensured that if an invalid value was read from the LiDAR, the previous value was used instead. This is done to minimise program disruption.

8.2 Evaluation

Recordings of the team's graded run can be found here.

8.2.1 Future Improvements

Location of Storage and Launching Subsystem

Currently the storage and launching subsystem are placed at the back of the robot. This was done as there was more space at the rear of the robot since the wheels are located at the front. However, this resulted in the robot having a large turning radius which was successfully addressed on the software side. With a successful evaluation, the team can determine that the launching and storage systems are working as desired. Hence, the team can work on reducing the size of the subsystems. The reduction in the size will thus allow for the subsystems to be attached to the front of the robot. By being attached at the front, the robot turning radius will hence be reduced, allowing for better manoeuvrability of the robot.

State Estimation with Kalman Filter

One of the factors leading to the poor implementation of navigating using waypoints and pathfinding is the inaccuracy of the robot movement. The inaccuracy in the robot movement can be improved through estimating the state (position and orientation) of the robot. This can be done by taking data from the robot's odometer and inertial measurement unit and combining it through Kalman Filter. Hence with a better estimation of the robot state, the accuracy in the robot's movement can be improved. thus improving navigation.

Optimising A for faster computation time*

Instead of using Python to compute the path, a C++ program can be written and compiled. This will greatly increase the speed of computation as it reduces the overhead found in Python. With a faster A* algorithm, the software can then compute paths at speeds where paths can be found in real-time. This will allow our navigation algorithm to be functional.

8.2.2 What went Well

High Reliability

Throughout the project, no electrical components were damaged. The only mechanical component that was damaged is the rack which was addressed promptly. This is a testament to the thoughtful engineering done by the group and can be attributed to the following steps done by the team. Components were tested extensively before integration and safety margins were factored in to reduce the chance of failure. Additionally, components specification was checked thoroughly to ensure that they do not exceed their limits.

Fastest Run

The team managed to complete mapping the given maze in 3 minutes and 24 seconds, which was the fastest timing for that maze. This can be attributed to extensive testing conducted by the team. The proportional controller values were fine-tuned through multiple runs, and the maximum angular and linear speed of the robot was determined. Proper mission planning led to time savings as the robot did

not stop at the loading zone until the map was fully complete. Additionally, by having the flexibility of choosing left or right wall tracking, the team was able to attempt the maze in either configuration. As such, the more optimum path can be chosen and run, leading to a faster mapping time.

Autonomous Completion
The robot managed to complete all tasks autonomously without interruption. This demonstrates the successful integration of individual tasks.

Acknowledgements

In a single semester, the team went through the process of designing, building, testing and evaluation. Multiple designs were iterated through and extensive testing was conducted before the graded run.

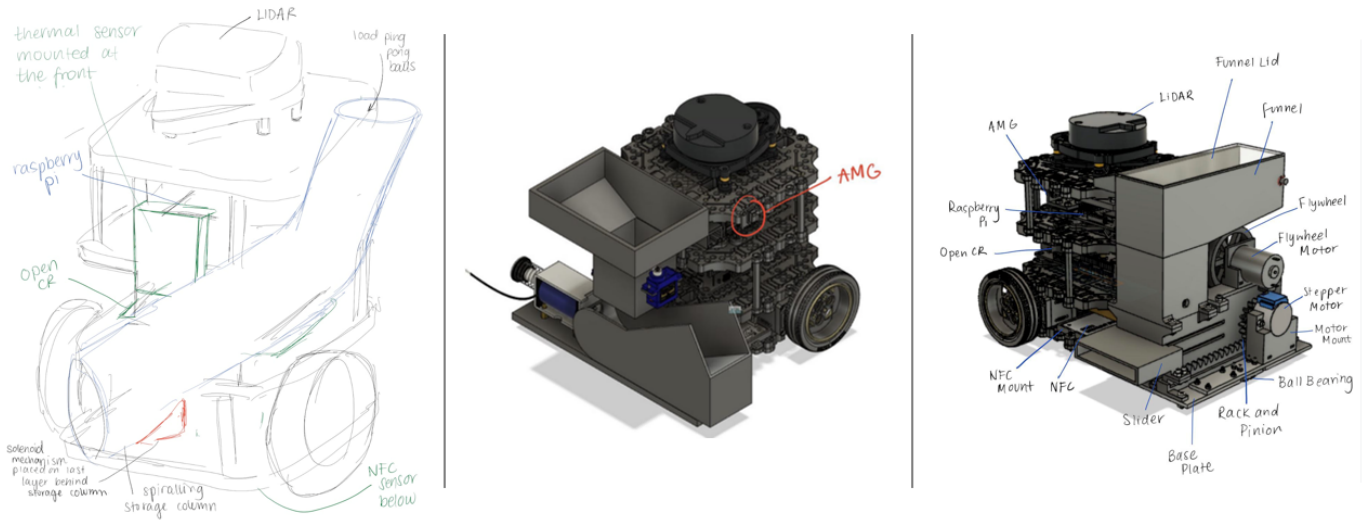


Figure: Iterations of the Team's Designs

It has been an intensive but fulfilling journey, which would not have been possible without the support of the following people: Ms Annie, Prof Soh, Prof Yen, Eugene, Royston, Keanne, Aurelian, Alvin and other teams taking the module. Hence, the team would like to express the team's deepest gratitude to the aforementioned people. The team will also like to thank Benjamin, Shaoliang and Kai Cong for assisting in the 3D printing of components.

Bibliography

1. M. Fazil, “Ros autonomous slam using randomly exploring random tree (RRT),” *Medium*, 15-Jan-2021. [Online]. Available: https://towardsdatascience.com/ros-autonomous-slam-using-randomly-exploring-random-tree-rrt-37186f6e3568. [Accessed: 23-Jan-2022].
2. “Robotis e-Manual ,” *Robotis* . [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#overview. [Accessed: 23-Jan-2022].
3. “tf Package ,” *ros.org*. [Online]. Available: http://wiki.ros.org/tf. [Accessed: 23-Jan-2022].
4. N. Benoit, M. Landergan, and O. Sanderson, rep.
5. “Near-field communication,” *Wikipedia*, 13-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Near-field_communication. [Accessed: 23-Jan-2022].
6. “Robotis eManual TurtleBot3,” *Robotis*. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_01/. [Accessed: 23-Jan-2022].
7. S. Guha and S. Kumar, rep.
8. S. Alamri, S. Alshehri, and W. Alshehri, International Journal of Mechanical Engineering and Robotics Research, rep., 2021.
9. J. Hrisko, “Thermal Camera Analysis with Raspberry Pi (AMG8833),” *MakerPortal* , 14-Jan-2021. .
10. S. Willner-Giwerc, “EV3 projectile launcher,” *LEGO Engineering*, 01-May-2020. [Online]. Available: http://www.legoengineering.com/ev3-projectile-launcher/. [Accessed: 23-Jan-2022].
11. “How do cam and follower mechanisms work?,” *YouTube*, 09-Mar-2020. [Online]. Available: https://www.youtube.com/watch?v=HsXWewecMLE. [Accessed: 23-Jan-2022].
12. icaresakr, “BALL3R, the Smart & Precise Lego Ball Shooting Robot,” *YouTube*, 15-Feb-2015. [Online]. Available: https://www.youtube.com/watch?v=cVMBcq6UY28. [Accessed: 23-Jan-2022].
13. “Single Flywheel,” *Flywheel - BLRS Wiki*. [Online]. Available: https://wiki.purduesigbots.com/hardware/shooting-mechanisms/flywheel. [Accessed: 23-Jan-2022].
14. “First Global Flywheel shooter tutorial,” *YouTube*, 15-Aug-2019. [Online]. Available: https://www.youtube.com/watch?v=An3xoJgk2uI. [Accessed: 23-Jan-2022].
15. captainranic, “How does a nerf gun work?,” *YouTube*, 28-Sep-2018. [Online]. Available: https://www.youtube.com/watch?v=N8JpePwvuHw. [Accessed: 23-Jan-2022].
16. S. Shah, “The dynamics of Ball Flight and design of a robotic ball shooter,” *Medium*, 08-Apr-2021. [Online]. Available: https://towardsdatascience.com/design-of-a-double-flywheel-variable-angle-ball-shooter-33221fa64866. [Accessed: 23-Jan-2022].
17. “Elastic catapult,” *Catapult - BLRS Wiki*. [Online]. Available: https://wiki.purduesigbots.com/hardware/shooting-mechanisms/catapult. [Accessed: 23-Jan-2022].

18. “Line Puncher - One Punch Bot ,” *Purdue Sigbots*. [Online]. Available: https://wiki.purduesigbots.com/hardware/shooting-mechanisms/linear-puncher. [Accessed: 23-Jan-2022].
19. “Pneumatics,” *Purdue Sigbots* . [Online]. Available: https://wiki.purduesigbots.com/hardware/pneumatics. [Accessed: 23-Jan-2022].
20. “Adafruit PN532 NFC/RFID Controller Shield for Arduino + Extras,” *Adafruit.com*. [Online]. Available: https://www.adafruit.com/product/789. [Accessed January 23, 2022.]
21. “2021,” *EG2310 Fundamentals of System Design*. [Online]. Available: https://blog.nus.edu.sg/eg2310/2021-2/. [Accessed: 23-Jan-2022].
22. “TurtleBot3 - Features,” *Robotis eManual*. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/features/. [Accessed: 23-Jan-2022].
23. “ROBOTIS TurtleBot3 Burger,” *Onshape*. [Online]. Available: https://cad.onshape.com/documents/2586c4659ef3e7078e91168b/w/14abf4cb615429a14a2732cc/e/36bfa8d5f9b09eac5379de52. [Accessed: 23-Jan-2022].
24. T. Harris, “How doorbells work,” *HowStuffWorks*, 10-Jan-2002. [Online]. Available: https://home.howstuffworks.com/home-improvement/repair/doorbell3.htm. [Accessed: 23-Jan-2022].
25. jbord39, “Solenoid to launch Tennis Ball,” *Electrical Engineering Stack Exchange*, 01-Jul-1964. [Online]. Available: https://electronics.stackexchange.com/questions/247949/solenoid-to-launch-tennis-ball/247955. [Accessed: 23-Jan-2022].
26. “Catapult,” *Wikipedia*, 22-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Catapult#:~:text=A%20catapult%20is%20a%20ballistic,energy%20to%20propel%20its%20payload. [Accessed: 23-Jan-2022].
27. Digital Manufacturing Solutions by CIM, “Launcher mechanism part 4 | first robotics competition - solid edge,” *YouTube*, 02-Feb-2020. [Online]. Available: https://www.youtube.com/watch?v=-XFIq_qI0bQ. [Accessed: 23-Jan-2022].
28. D. Shah, “Build your own thermal camera with Raspberry Pi and CJMCU-8833 thermal image array temperature sensor,” *Circuit Digest*, 14-May-2021. [Online]. Available: https://circuitdigest.com/microcontroller-projects/thermal-camera-with-raspberry-pi-and-cjmcu-8833-thermal-image-array-temperature-sensor. [Accessed: 23-Jan-2022].
29. A. Gregory, “Read RFID and NFC tokens with Raspberry Pi: Hackspace 37,” *Raspberry Pi*, 18-Sep-2021. [Online]. Available: https://www.raspberrypi.com/news/read-rfid-and-nfc-tokens-with-raspberry-pi-hackspace-37/. [Accessed: 23-Jan-2022].
30. “Lidar,” *Wikipedia*, 22-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Lidar. [Accessed: 23-Jan-2022].
31. National Oceanic and Atmospheric Administration - US Department of Commerce, “What is Lidar,” *NOAA's National Ocean Service*, 01-Oct-2012. [Online]. Available: https://oceanservice.noaa.gov/facts/lidar.html#:~:text=Lidar%2C%20which%20stands%20for%20Light,variable%20distances)%20to%20the%20Earth.&text=A%20lidar%20instrument%20principally%20consists,and%20a%20specialized%20GPS%20receiver. [Accessed: 23-Jan-2022].

32. “Robot operating system,” *Wikipedia*, 07-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Robot_Operating_System. [Accessed: 23-Jan-2022].
33. “Wiki,” *ros.org*. [Online]. Available: http://wiki.ros.org/ROS. [Accessed: 23-Jan-2022].
34. “SLAM,” *Robotics eManual*. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/slam/#run-slam-node. [Accessed: 23-Jan-2022].
35. “Maze-solving algorithm,” *Wikipedia*, 10-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Maze-solving_algorithm#:~:text=A%20maze%2Dsolving%20algorithm%20is,for%20the%20solving%20a%20maze.&text=Mazes%20containing%20no%20loops%20are,closely%20related%20to%20graph%20theory. [Accessed: 23-Jan-2022].
36. “Projectile of a Trajectory: With and Without Drag,” *Desmos*. https://www.desmos.com/calculator/on4xzwtdwz.
37. J. Sluka, “A PID controller for Lego Mindstorms Robots,” *PID Controller For Lego Mindstorms Robots*, 05-Oct-2009. [Online]. Available: https://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html. [Accessed: 20-Apr-2022].
38. M. Finkelstein and B. Hunte, “Wall Following with Collision Avoidance and Mapping Using a Laser Range Finder,” *A. James Clarke School of Engineering* . [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.503.1317&rep=rep1&type=pdf. [Accessed: 20-Apr-2022].
39. “13. Api - input devices — gpio zero 1. 6. 2 documentation.” [Online]. Available: https://gpiozero.readthedocs.io/en/stable/api_input.html?highlight=lightsensor#lightsensor-ldr [Accessed Apr. 21, 2022].

Appendix A: Preliminary Mechanical Design

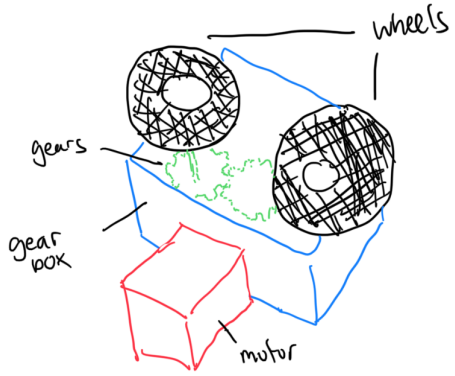


Figure 1: Flywheel Design

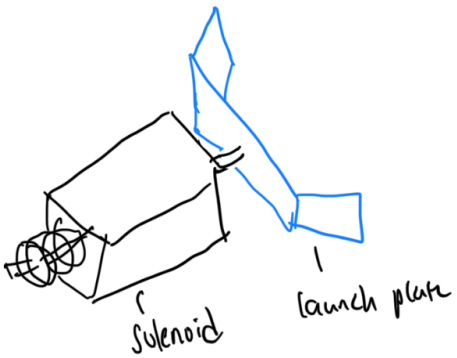


Figure 2: Using Solenoid as a kicker

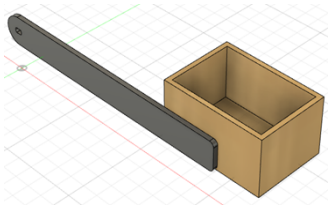


Figure 3: Catapult Arm

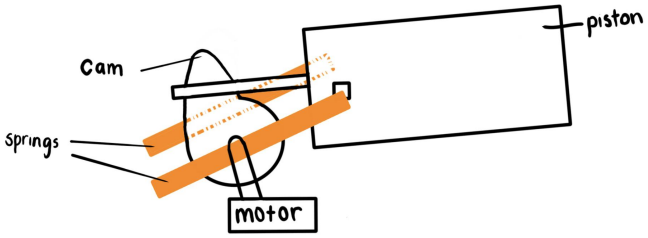


Figure 4: Cam based shooting mechanism

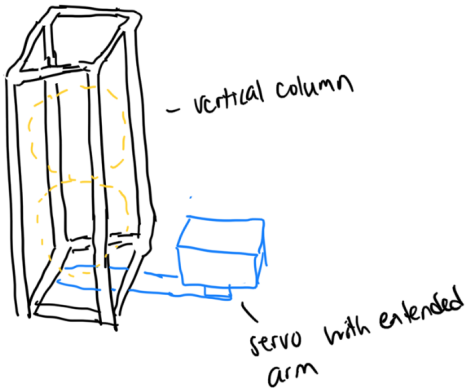


Figure 5: Vertical Storage System

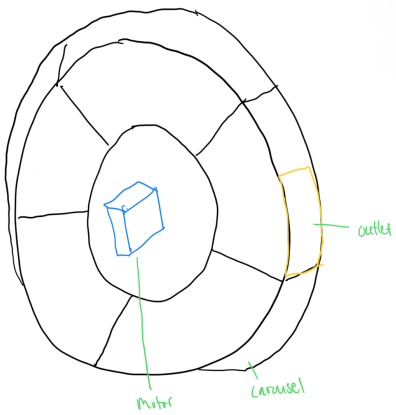


Figure 6: Carousel Storage System

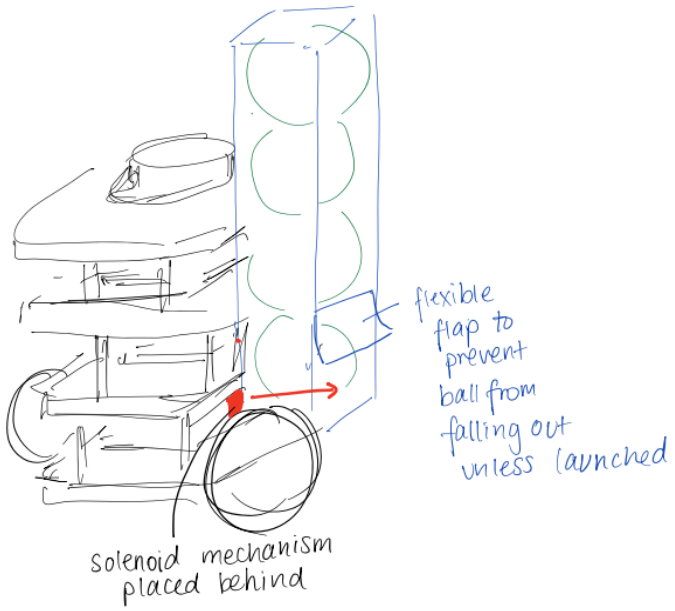
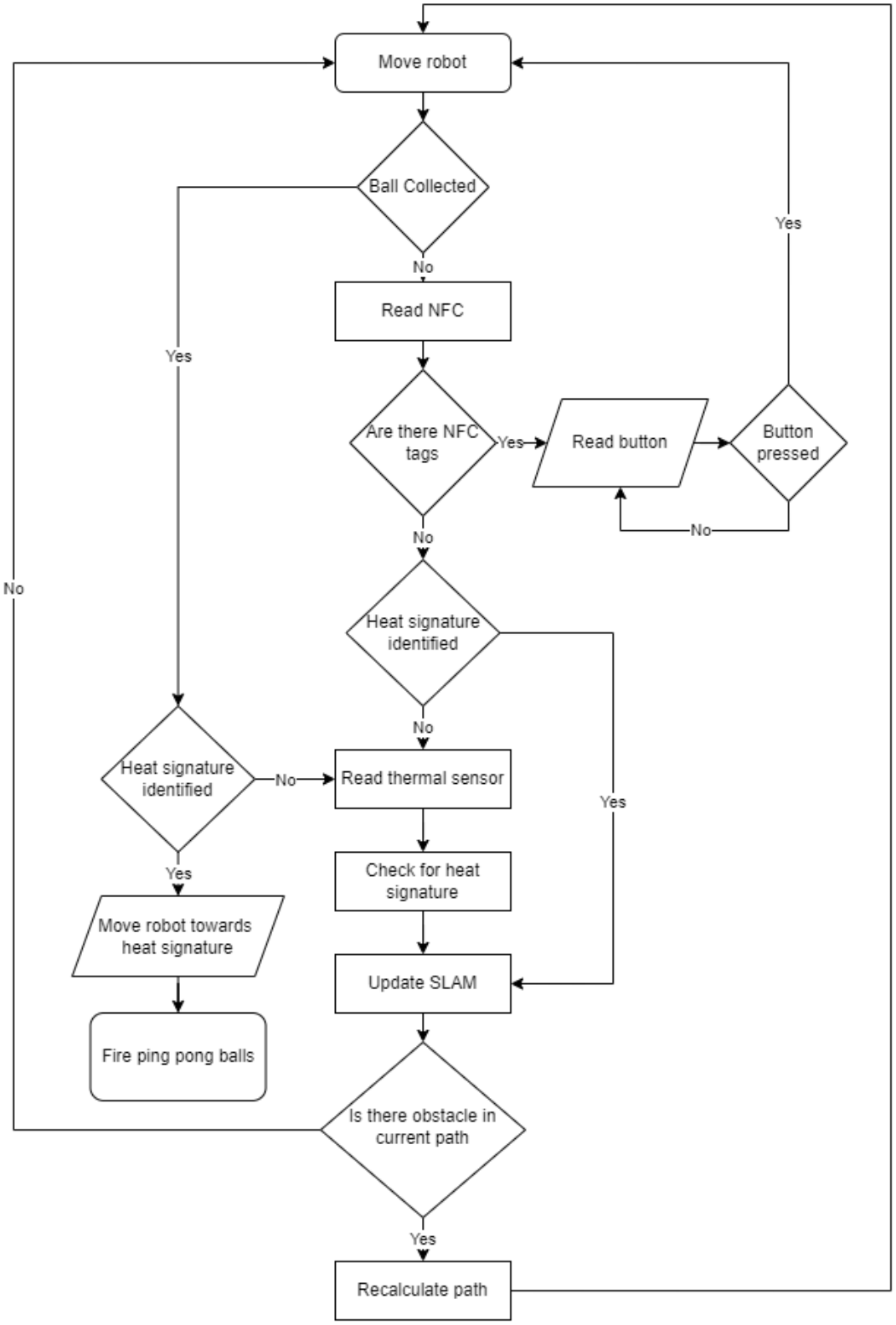
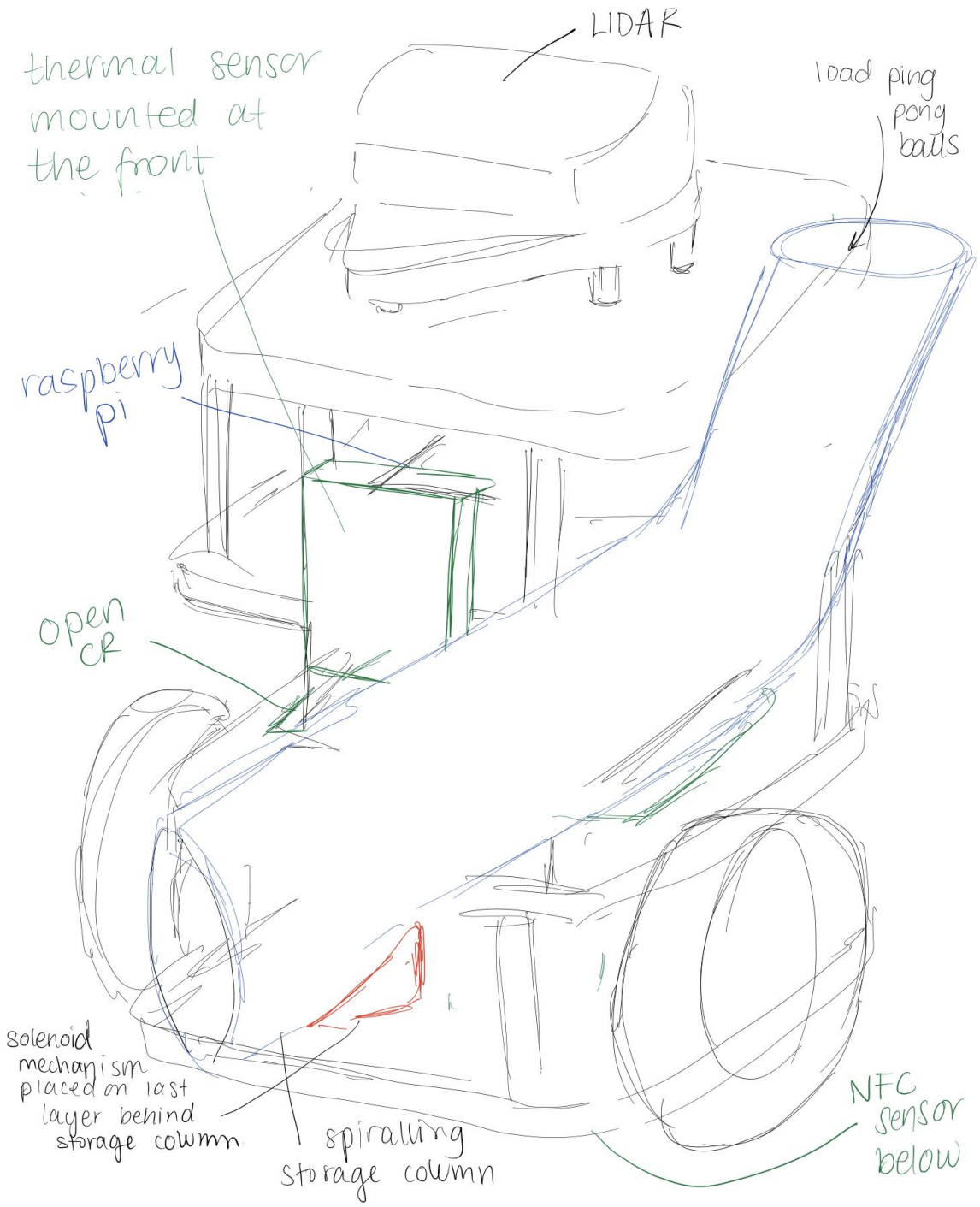


Figure 7: Using a Vertical Storage System

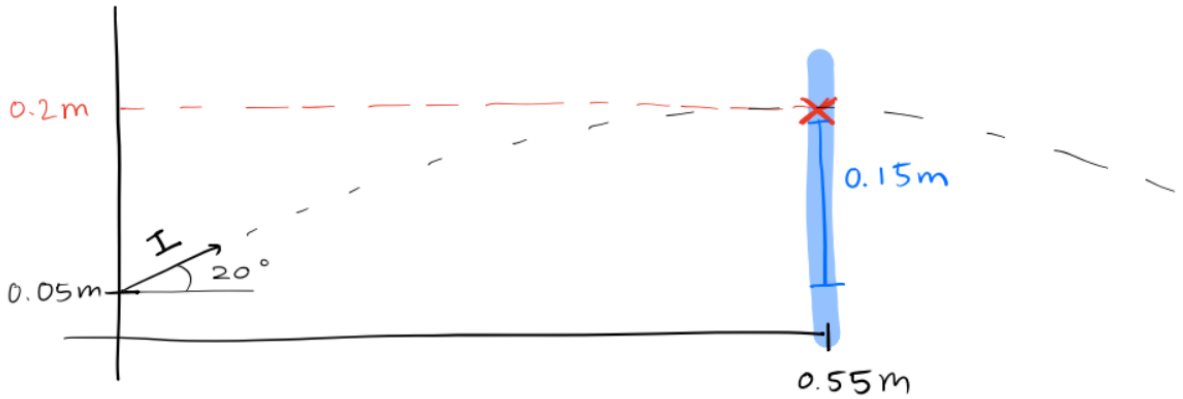
Appendix B: Preliminary Program Flow

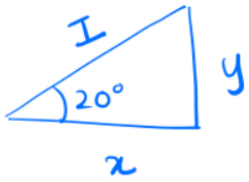


Appendix C: Preliminary Robot Design



Appendix D: Ball Trajectory Calculation





I = initial velocity

y = initial velocity in y plane

= I sin 20°

x = initial velocity in x plane

= I cos20°

	X-Axis	Y-Axis
Distance (s)	0.55	0.15
Initial Velocity (u)	I * cos(20)	I * sin(20)
Final Velocity (v)	I * cos(20)	0
Acceleration (a)	0	-9.81

$v^2 = u^2 + 2as$

$0 = I^2 sin(20) + 2(-9.81)(0.15)$

$294.3 = 0.117 * I^2$

$I^2 = 25.159$

$I = 5.016\, m/s$

Initial velocity required will be slightly more than 5m/s.

Appendix E: Solenoid Components Calculation

The team plan to overvolt a solenoid rated at 24V to achieve higher launching speed. The team will overvolt it to 48V at the expense of its lifespan. Since it has an average lifespan of 0.5 million on and off cycles, which the team does not need, it is thus safe and reliable for us to overvolt.
Calculating initial kinetic energy of the ping pong ball,
$E = ½ * C * v^2 = ½ * 0.003 * 25 = 0.0375J$
Assuming no energy lost, at 48V a capacitor of 32.5uF is needed.

However, assuming an energy loss of 90%, a capacitor of 325uF is needed. Correspondingly, the discharge time constant will be 5.5ms, assuming the team is using a solenoid rated for 24V and 1.4A which has an internal resistance of 17 ohms.

Appendix F: Solenoid Test Calculation

Test	Maximum distance travelled by ball/cm
1	16.3
2	18.8
3	17.1

$Average\ distance\ travelled\ by\ ball\ =\ 17.4cm$

$Angle\ of\ incline\ = 30^{\circ}$

$F_{g\ on\ ball} = m_{ball} * g$

$F_{g\ on\ ball\ parallel\ to\ slope} = m_{ball} * g * sin(30)$

$a = g * sin(30) = 4.9ms^{-2}$

$u = 1.3ms^{-1}$

Given the initial velocity, the distance the ball can travel horizontally:

$u_y = u * sin(30) = 0.65ms^{-1}$

$t = 2 * \frac{v-u}{a} = 2 * \frac{-0.65}{-9.81} = 0.132s$

$s_x = u * cos(30) * t = 0.65 * cos(30) * 0.132 \approx 0.074m = 7.4cm$

The distance the ball travels horizontally, s_x is an underestimate of the actual distance the ball will travel when fired from the robot's solenoid launcher. This is due to the energy loss to friction while the ball is on the slope during the experiment.

Even so, the estimate of the distance should not be too far off from the actual distance as the energy loss to friction is not very large due to the low mass of the ball and the short distance the ball travelled.

Hence, the team can conclude that the horizontal distance the solenoid is able to launch the ball falls short of the team target distance of at least 30cm.

Appendix G: Power Consumption Tests

Power Consumption of Turtlebot Starting Up

	Typical	Surge
Voltage	11.109 V	11.109 V
Current	0.620 A	0.939 A
Power	11.109 * 0.620 = 6.8876 W ≈ 6.89 W	11.109 * 0.939 = 10.4314 W ≈ 10.43 W

Power Consumption of Turtlebot Powered On (Standby/Operation)

	Idling	Running Auto_Nav
Voltage	11.109 V	11.109 V
Current	0.580 A	0.660 A
Power	11.109 * 0.580 = 6.4432 W ≈ 6.44 W	11.109 * 0.660 = 7.33194 W ≈ 7.33 W

Estimation of Battery Life

Battery Specification

Voltage: 11.1V
Capacity: 1800mAh
Discharge Rate: 35C

Amount of Energy Stored: 11.1 * 1.8 = 19.98 Wh
Max Safe Current Draw: 1.8 / (1/35) = 63 A

Power used for 2 minutes (1 minute typical, 1 minute surge) start up:
6.89 * (1/60) + 10.43 * (1/60) = 0.28867 Wh
Energy left after start up: 19.98 - 0.28867 = 19.69133 Wh

Using the typical operating power of 7.33 W with a safety margin of 20%:
(19.69133 * 0.8) / 7.33 = 2.1491 Hours ≈ 2 Hours 9 Mins

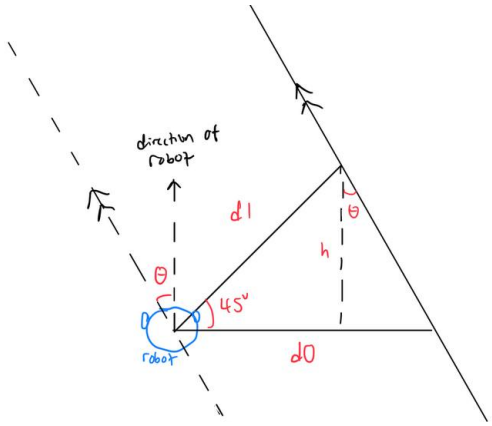
Draining only 70% of battery:
2.1491 * 0.7 = 1.5044 Hours ≈ 1 Hour 30 Mins

Assumptions

1) There is no huge sudden surge in power usage
2) The battery capacity is still at its maximum and hasn't degraded over the years of usage
3) Components which are only activated for a small number of times like the flywheel and stepper motor are catered for in the safety margin of 20%

Appendix H: Calculations for triangle wall tracking

Part 1: Correction of heading of robot



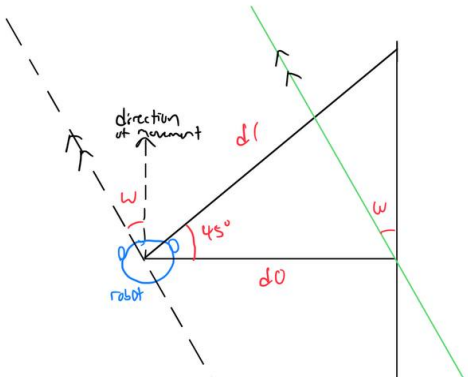
$h = d_1 sin(45)$

$projection\ of\ d_1\ onto\ d_0 = d_1 cos(45)$

$\Theta = tan^{-1}(\frac{d_1 cos(45) - d_0}{d_1 sin(45)})$

$angular\ velocity\ = k_\theta * \Theta$ where k_θ is determined experimentally

Part 2: Correction of distance of robot from the wall



Let x be the distance the target tracking distance of the robot from the wall

d_0 is the shortest distance from the centre of the robot to the wall

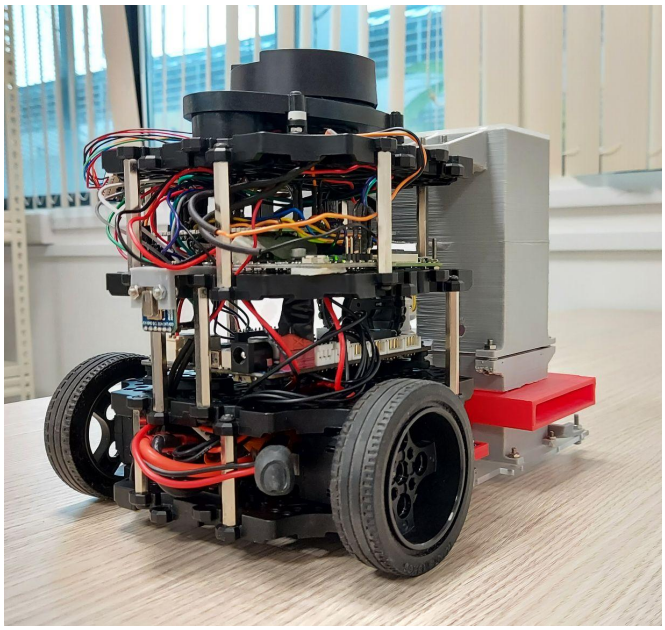
$error_{dist} = x - d_0$

$\omega = k_{dist} * error_{dist}$ where k_{dist} is determined experimentally

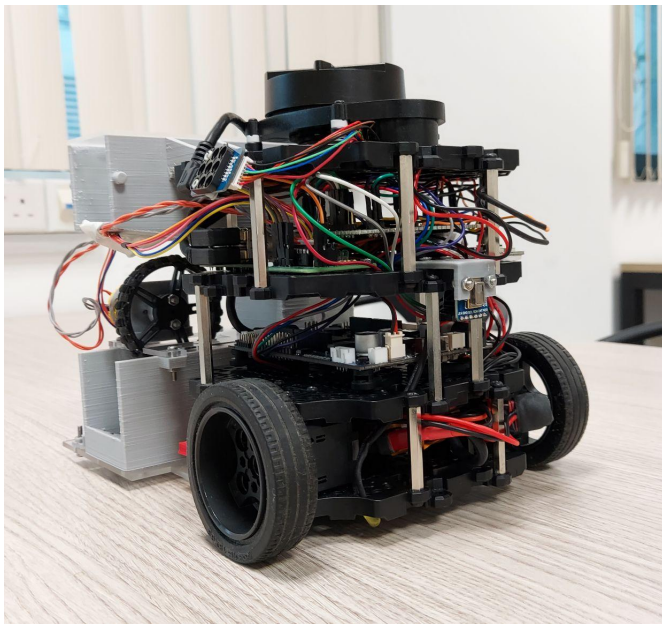
Combining the results from both parts,

$angular\ velocity\ = k_\theta * (\Theta + \omega)$

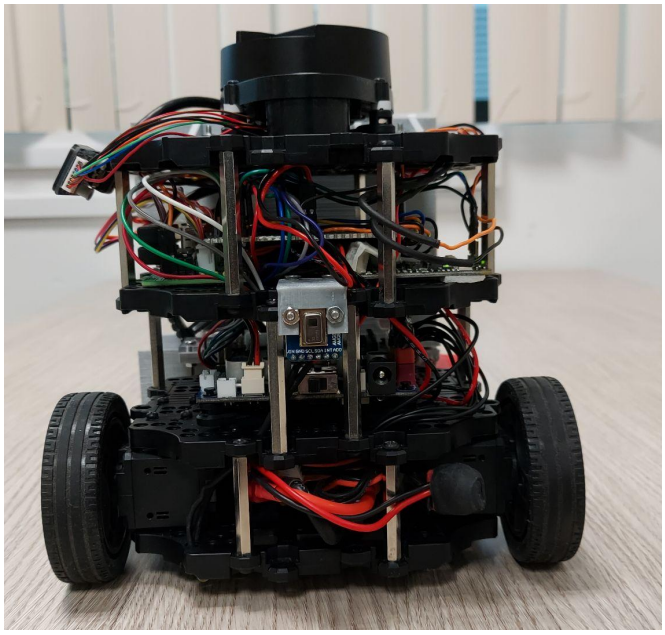
Appendix I: Photograph of Final Physical Robot



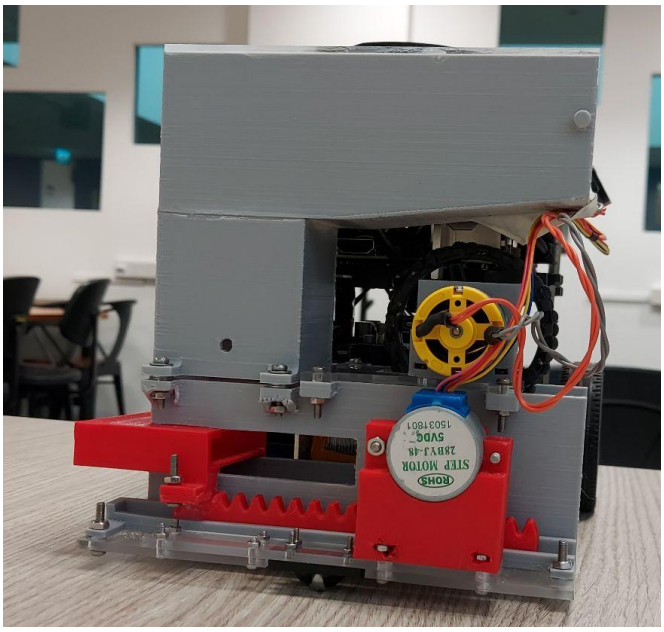
Front Left View of Robot



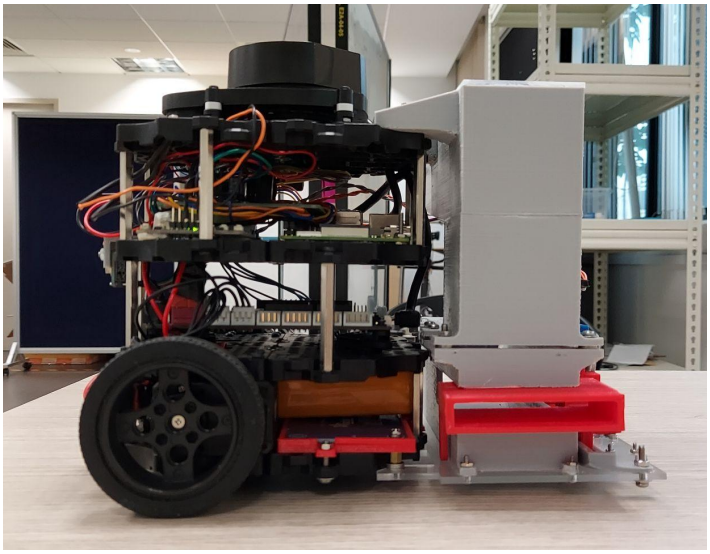
Front Right View of Robot



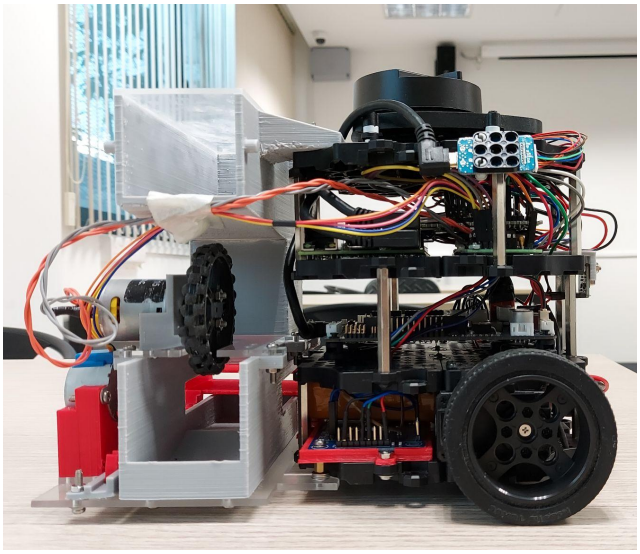
Front View of Robot



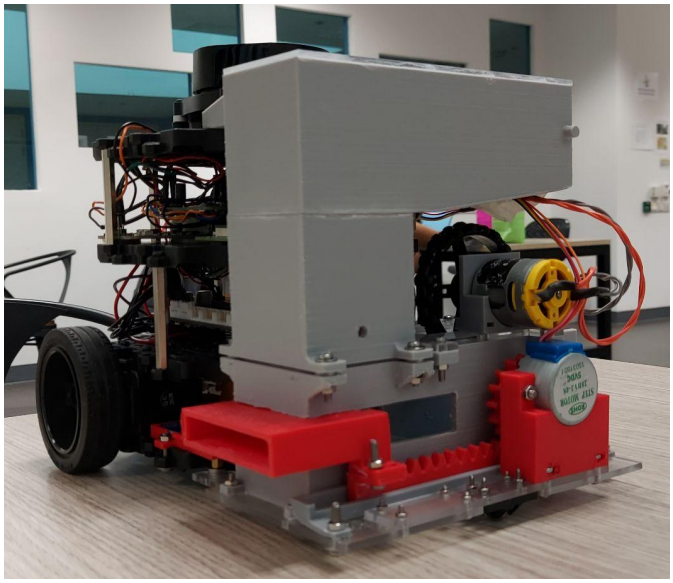
Rear View of Robot



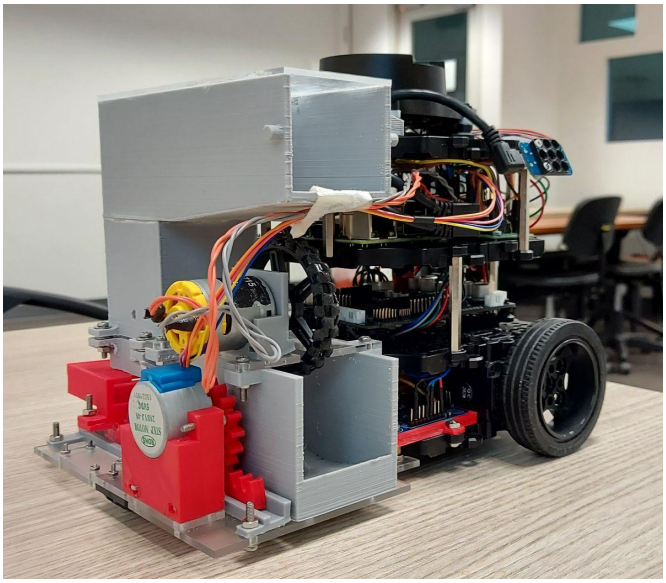
Left View of Robot



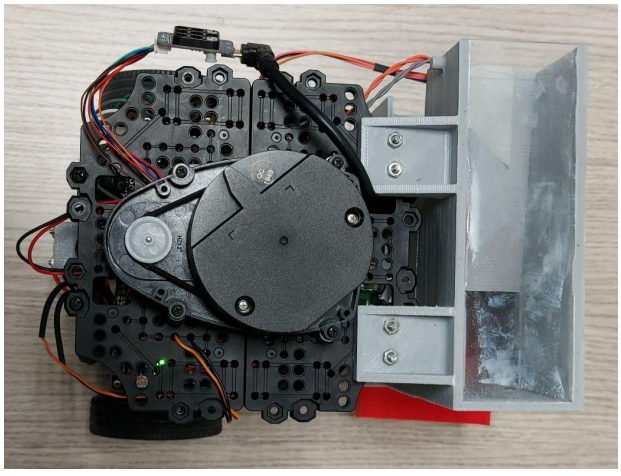
Right View of Robot



Rear Left View of Robot

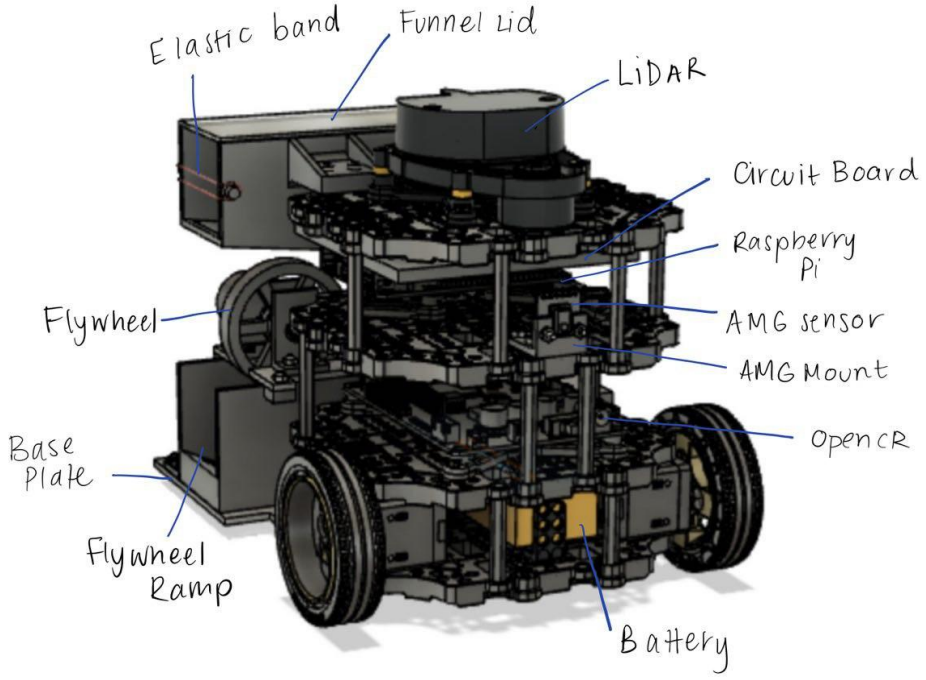


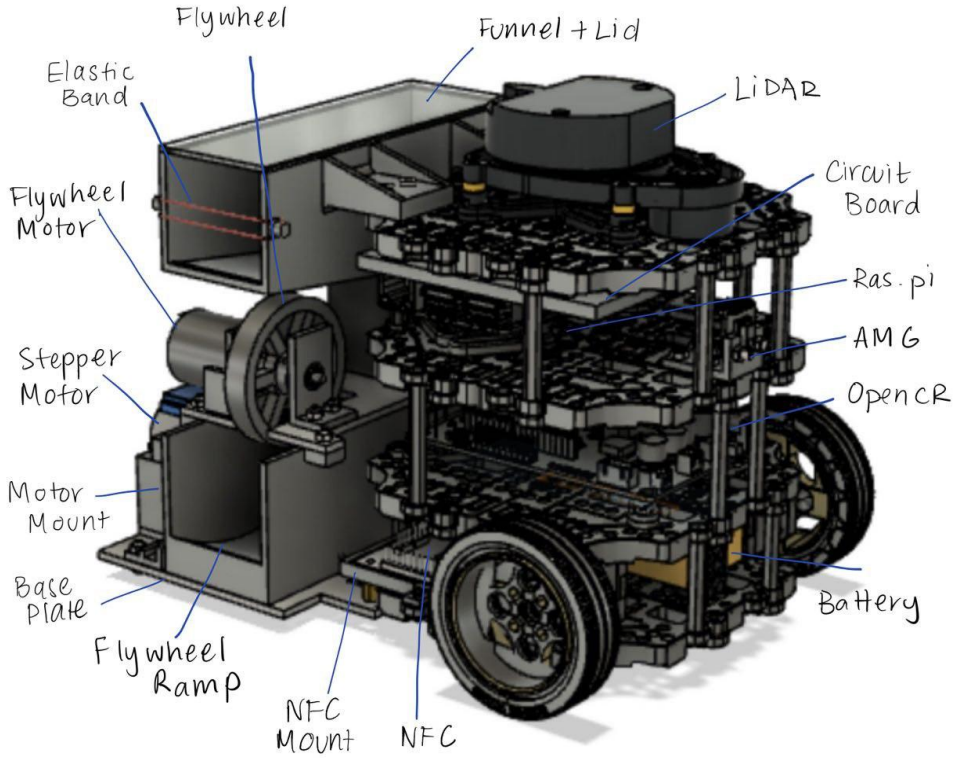
Rear Right View of Robot

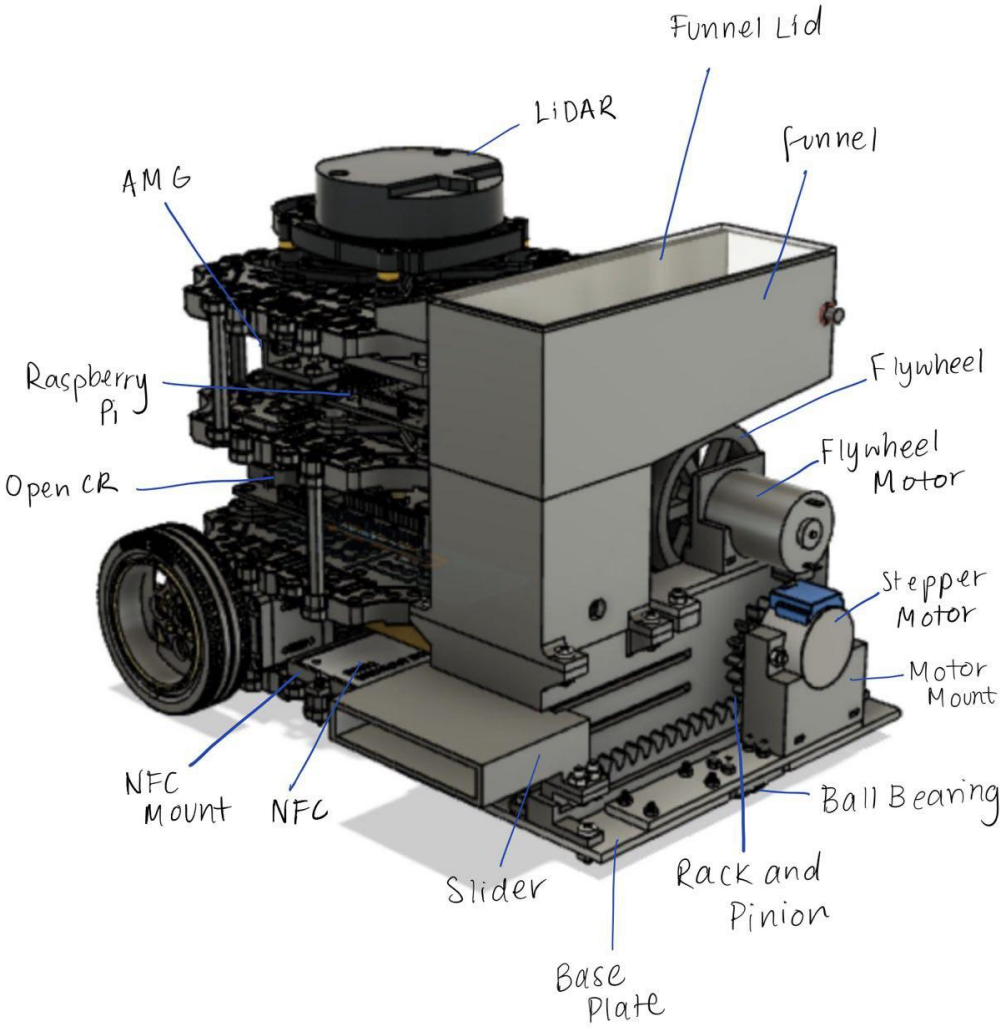


Top View of Robot

Appendix J: Labelled CAD of Robot



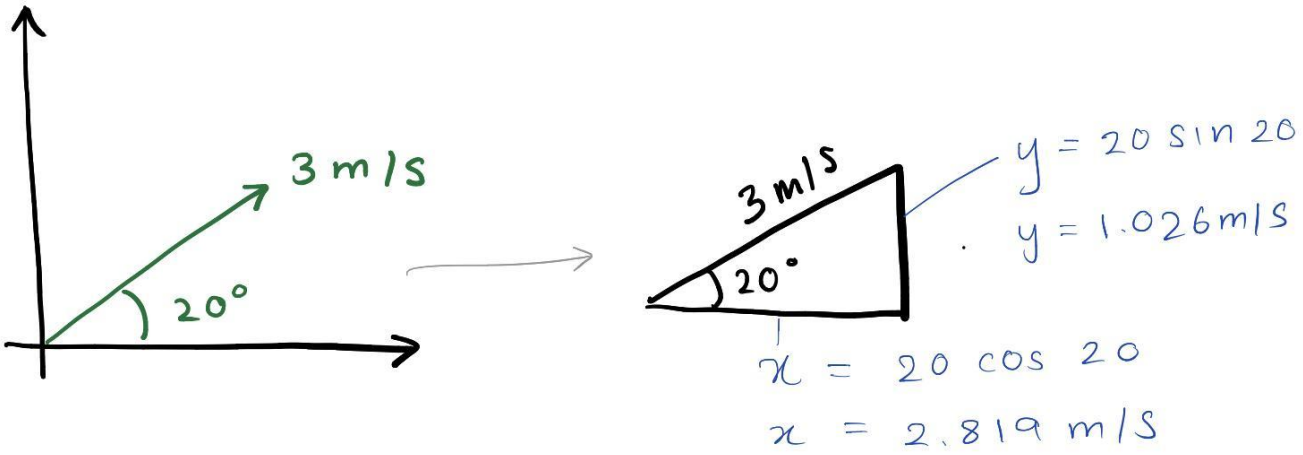



Funnel Lid
LiDAR
Funnel
AMG
Raspberry Pi
Flywheel
Flywheel Motor
Open CR
Stepper Motor
Motor Mount
NFC Mount
NFC
Ball Bearing
Slider
Rack and Pinion
Base Plate

Appendix K: Flywheel Calculations

Flywheel Diameter: 6 cm
Angle of Launch: 20 degrees
Rotational Velocity: 10 000 rpm
Velocity of Flywheel: 61 m/s
Mass of Ping Pong Ball: 2.7 g
Diameter of Ping Pong Ball: 40 mm
Surface Area of Ping Pong ball: 0.00126 m^2

Taking velocity of ball to be 3 m/s (95% loss)



	X - axis	Y - axis
Distance (m)	**0.296**	-
Initial Velocity (m/s)	2.819	1.206
Final Velocity (m/s)	2.819	0 (at the peak of the projectile - halfway)
Acceleration (m/s^2)	0	-9.81
Time	0.105	

If 0.296 m is half of the horizontal distance travelled, the ball can be launched upto 0.592 m on the projectile.

Through calculations, it is determined that even with an estimated 95% loss, the flywheel is able to launch and hit the target. The huge estimated loss was overestimated to have leeway to control the launcher speed.

Appendix L: Links to Other Resources

Github Repository: https://github.com/eg2310g8/r2auto_nav

General
Loading Manual:
https://github.com/eg2310g8/r2auto_nav/blob/main/Documentations/Loading_Manual_for_TAs.pdf
Detailed Bill of Materials:
https://github.com/eg2310g8/r2auto_nav/blob/main/Documentations/detailed_bill_of_materials.pdf
End User Documentation:
https://github.com/eg2310g8/r2auto_nav/blob/main/Documentations/G8_End_User_Document.pdf
Recordings of Graded Run:
https://photos.google.com/share/AF1QipMOHCC4Hmy2iMv2cuJiwt46ZaMUjUX0MDv22KL1oktZ5YMMdFfMDikiKv0L4tRCtg?key=cUpqMlcxdmlxRlFrZUQwS3BXZ1cxc0tzVDR4VG5B
Youtube Playlist:
https://youtube.com/playlist?list=PLSZRJ42bi6Z_z2Q0XyNdcpAQX7ErHS6G8

Mechanical
Assembly Documentation:
https://github.com/eg2310g8/r2auto_nav/blob/main/Documentations/detailed_assembly_document.pdf
STL & DXF Files:
https://github.com/eg2310g8/fabrication

Electrical
Perfboard Schematics:
https://github.com/eg2310g8/r2auto_nav/blob/main/Documentations/perfboard_schematics.png
Electronic System Architecture:
https://github.com/eg2310g8/r2auto_nav/blob/main/Documentations/electronic_system_architecture.png

Software
Software Block Diagram:
https://github.com/eg2310g8/r2auto_nav/blob/main/Documentations/software_block_diagram.png