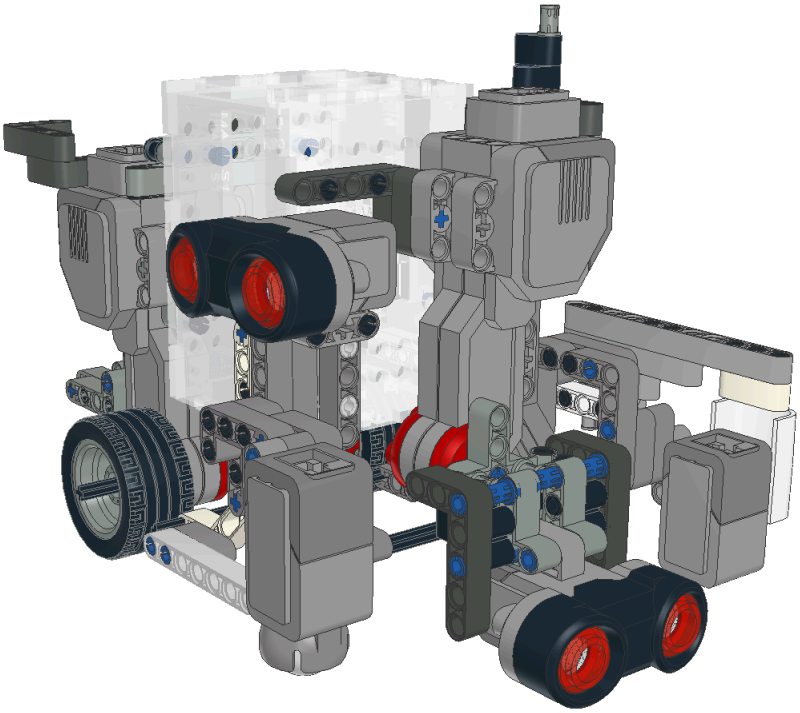
# Final Project Report

## Design Principles and Methods

Version 3.0 ECSE 211

Dec. 4, 2024 McGill University

William Blackmore – Marrec Bois – Garrett Woodson – Karim Gaafar – Nicolas Dolgopolyy
Katrina Panwar



**Team 9: Marwan-Bot**

*Named after the professor who reminded us that even the most challenging Engineering problems can be solved by following the process.*

McGill | Faculty of Engineering

# Table of Contents



# Nomenclature

| **Acronym** | CSR | CSL | USR | USF |
|---|---|---|---|---|
| **Sensor** | Color Sensor Right | Color Sensor Left | UltraSonic Right | UltraSonic Front |

| **Acronym** | LMR | LML | LMF |
|---|---|---|---|
| **Motor** | Large Motor Right | Large Motor Left | Large Motor Front |

# I. Introduction

Gather round, for in this report lies the saga of the Marwan-Bot's design and assembly. Dubbed in homage to our beloved professor, it is a true hero of park dog poop cleanup initiatives.

## A. Team Capabilities and Organization

This section outlines the diverse capabilities of our engineering design team, detailing the strengths and past experiences which we leveraged to drive our team organisation. While developing this document, we considered each team member's unique background, ensuring their roles align with their area of expertise.

| Name | Roles | Responsibilities | Reason for Member's Role |
|---|---|---|---|
| William Blackmore | **Lead Documentation**<br><br>-Software Engineer | Maintaining comprehensive documentation, coordinating team processes, and ensuring workflow efficiency. | Proficient Python developer with a portfolio of side projects and strong documentation skills. |
| Nicolas Dolgopolyy | **Lead Hardware**<br><br>-Software Engineer | -Leading design iterations in hardware development<br>-Managing hardware testing<br>-Program certain functions | Extensive experience in robotics, R&D, with a strong proficiency in algorithm design in robotics. |
| Garrett Woodson | **Project Manager**<br><br>-Software Engineer | -Manages Gantt Chart and design processes<br>-Maintain cross-team communication<br>-Help out on coding | Strong experience in embedded software, low-level programming. Leadership experience acquired through managing McGill's 3D printing service. |
| Karim Gaafar | **Lead Draughtsman**<br><br>-Hardware Engineer<br>-Test Engineer | -Model system using CAD<br>-Develop hardware systems<br>-Implementing structured testing<br>-Support documentation | Foundational skills in electrical engineering and the hardware aspect of robotics. Strong skills in electrical circuit design and experience with CAD softwares. |
| Marrec Bois | **Lead Software**<br><br>-Testing Engineer | Overseeing software development, ensuring timely task completion, and guiding the team effectively. | Extensive Python and web development expertise, with leadership experience and a focus on algorithmic thinking. |
| Katrina Panwar | **Testing Lead**<br><br>-Documentation | -Coordinate testing efforts<br>-Systematic testing of software and hardware<br>-Support Documentation | Expertise in unit and integration testing, with strong organizational skills and a background in web development. |

Table 1. Team Roles and Responsibilities

## B. Team Availability and Team Contract

Understanding the availability of each team member is critical for effective planning and task allocation. This section details the availability schedules of all team members, providing a clear view of when individuals are accessible for collaboration.

Throughout this project, each team member is committed to working an average of 9 hours per week, as stipulated in the client contract. Team members coordinated their schedules to ensure availability for both weekly internal meetings as well scheduled meetings with our assigned Senior Engineer.

In order to effectively align our schedules, we leveraged the online tool "when2meet" to drive a consensus on the best meeting time that accommodated everyone's availability efficiently (See

Appendix Table A.1). We decided to have bi-weekly full-team meetings during the pre-allocated ECSE 211 class timeslots, where we provided updates on our assigned tasks. All deviations from the standard schedule were communicated to the rest of the team via the corresponding channel in Slack.
The following table shows the additional availability specific to each team member. Note that our set **ECSE 211 timeslots was the default availability** for everyone.

| Team Member | Availability |
|---|---|
| William | ➢ Weekends<br>➢ Monday afternoons, Tuesdays, and Friday mornings |
| Nicolas | ➢ Weekends (except Nov. 9th and Nov 16-17th)<br>➢ Tuesdays, Thursdays, and Friday mornings |
| Garrett | ➢ Weekends (except Nov. 2-3rd, Nov. 9-10th, and Nov. 22-24th)<br>➢ Monday afternoons, Tuesdays, and Friday mornings |
| Karim | ➢ Weekends<br>➢ Monday-Thursday (12-2pm), and Friday mornings |
| Marrec | ➢ Weekends<br>➢ Monday afternoons, Tuesdays, and Friday mornings |
| Katrina | ➢ Weekends (except Nov. 9th)<br>➢ Monday afternoons, Tuesdays, and Friday mornings |

Table 2. Team Availability Summary

Furthermore, our team contract and shared goals form the foundation of our collaboration, setting clear expectations and aligning our efforts toward a common vision. Drafted during our initial meeting, this contract between members reinforced our adherence to both the agreed upon schedule as well our commitments to one-another (see Appendix Doc A.1). This document details a set of guidelines each member will follow and the shared goals we will work towards. We regarded responsiveness and accountability as paramount values, as they foster trust among team members and enable us to address challenges while maintaining momentum towards our objectives.

## C. Project Management

Project management was pivotal in steering our team's progress, ensuring that we met our milestones, allocated resources efficiently, and adapted to challenges effectively. This section provides an overview of the milestones we defined, our strategies for monitoring progress, and adjustments made to our budget.

**Milestones and Planning**
At the project's inception, we established five key milestones to guide us through the design, development, and testing phases of our robot. Each milestone was targeted for completion at the end of each week:

- **Week 1 Initial Milestone:** Initial Hardware and Software Plan Created
- **Week 2 Initial Milestone:** First Semi-Functional Product Built
- **Week 3 Initial Milestone:** Prototype Finalized and Optimization Begun
- **Week 4 Initial Milestone:** First Report Draft Complete and Robot Demo Ready
- **Week 5 Initial Milestone:** Report Finalized and Demo Performed

For each milestone, we created a comprehensive list of dependencies and assigned them to team members based on expertise and estimated time requirements. Figure 1 shows the time allotted to different task categories such as hardware, software, and project management. As can be seen from the chart, testing, documentation, and software were the most expensive categories in terms of budget usage. The multidisciplinary composition of our team allowed us to allocate tasks

efficiently—for instance, Karim and Nic focused on mechanical design, Will, Marrec, and Garrett developed software algorithms, and Katrina led testing.

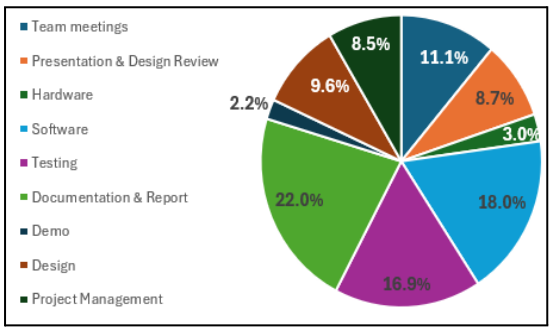



Figure 1. Time Distribution per Task

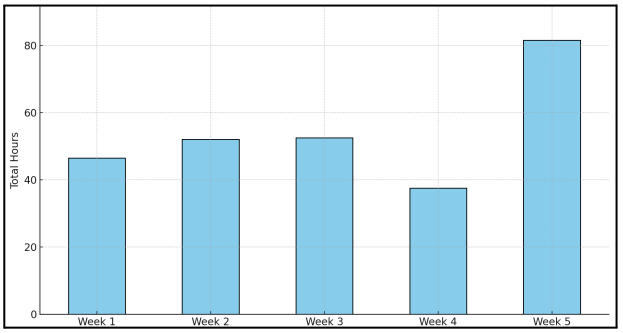



Figure 2. Total Time Spent by Weekly Milestone

**Progress Monitoring and Team Coordination**

To ensure smooth collaboration, we established a team contract outlining our communication protocols, decision-making processes, and conflict resolution strategies. We agreed to meet twice weekly:

- **Monday Meetings:** Allocate tasks for the week in line with the project budget and Gantt chart.
- **Friday Meetings:** Review progress on assigned tasks and address any obstacles encountered.

Throughout the week, we utilized Slack for real-time communication, enabling prompt discussions and quick resolution of issues. This consistent communication was crucial for maintaining alignment and momentum.

**Budget Adjustments and Scheduling**

Initially, we developed our project budget and Gantt chart based on estimated times for each task (see Appendix Figure A.1 and Figure A.2). However, as the project progressed, we encountered unforeseen dependencies—such as hardware modifications required to implement specific software algorithms—that necessitated adjustments to the project budget and milestones. We discovered that we overestimated the time needed for early tasks like requirement writing and constraint review. Conversely, we underestimated the complexity and time required for later tasks like developing the cube and water avoidance systems. This misjudgment occurred because we didn't fully anticipate the intricacies involved in integrating hardware and software components. To adapt, we updated our Gantt chart and budget to reflect actual time expenditures and resource needs (see Appendix A.3 and A.4). We reallocated time from overestimated tasks to those requiring additional effort, ensuring that we stayed on track without compromising quality. Table 3 shows the week by week time spent vs time allotted for each team member. Furthermore, the project milestones were updated to reflect the team's true progress since over the course of the project the initial milestones could not be met due to the unforeseen dependencies described above. The updated project milestones were as follows:

- **Week 1 Updated Milestone:** Initial Hardware and Software Plan Created
- **Week 2 Updated Milestone:** Subsystems are Functional and Unit Testing Begun
- **Week 3 Updated Milestone:** Subsystems Integrated and Integration Tests Begun
- **Week 4 Updated Milestone:** Prototype Finalized and System Tests Begun
- **Week 5 Updated Milestone:** Robot Demo Ready and Report Complete

| **Member** | Week 1 | Week 2 | Week 3 | Week 4 | Week 5 | **Total By Member** |
|---|---|---|---|---|---|---|
| William | 6.5 / 8.25 | 4.5 / 10.75 | 7 / 7.5 | 10 / 8 | 23 / 13 | **51 / 45** |

| | | | | | | |
|---|---|---|---|---|---|---|
| Nicolas | 7.5 / 9.75 | 3 / 10.75 | 3 / 6 | 7 / 5.5 | 21 / 13 | **41.5 / 45** |
| Garrett | 5.5 / 6.25 | 3 / 4.25 | 10 / 10 | 9 / 8 | 22.5 / 16.5 | **49 / 45** |
| Karim | 3.75 / 6.75 | 5.25 / 10.75 | 6.75 / 8.5 | 10.25 / 5.5 | 16 / 13.5 | **42 / 45** |
| Marrec | 7.5 / 8.75 | 4.5 / 7.75 | 8.5 / 10 | 12.5 / 6.5 | 20 / 12.5 | **53 / 45** |
| Katrina | 4.5 / 6.75 | 4 / 7.75 | 4 / 10.5 | 5.5 / 5.5 | 20 / 14.5 | **38 / 45** |
| **Total by Week** | **35.25 / 46.5** | **24.25 / 52** | **39.25 / 52.5** | **54.25 / 37.5** | **122.5 / 81.5** | **274.5 / 270** |

Table 3. Time Spent vs Time Budgeted (in hours)

We over budgeted time for Week 2 while underestimating the time needed in the final weeks, which proved to be a mistake. In Week 2, we allocated more time than necessary for tasks like setting up the semi-functional product, assuming they would be more complex, which left some team members with extra capacity.

In contrast, the last few weeks were under budgeted due to an underestimation of the effort required for tasks such as optimizing cube detection and water avoidance systems. In hindsight, we should have redistributed unused time from Week 2 to the later stages and reviewed our time allocation more frequently to address these imbalances earlier. This approach would have ensured a more balanced workload and reduced pressure during the final stages. Ultimately, we finished this project having utilized 102% of our allotted time budget. The team went over budget slightly due to unforeseen circumstances in the final week of the project. Namely, an update made to the grading rubric of the project demo led Garrett, the project manager, to organize an impromptu team meeting to discuss if the updated rubric should affect the previously agreed upon strategy for the demo. This meeting hadn't been included in the original budget and caused the team to go slightly over budget.

# II. Specifications

## A. Problem Scope

Many cities lack adequate facilities to promote responsible dog ownership. As a consequence, the absence of these amenities leads to unattended dog waste in public areas, causing public disapproval of dog owners. To resolve this issue, our design team is tasked with developing a prototype of an autonomous robot to clean dog waste in public parks. The scope of our project includes:

- **Navigation:** Designing the robot to efficiently traverse park terrains while avoiding obstacles and bodies of water.
- **Waste Collection:** Implementing a reliable system to collect and store multiple pieces of dog waste.
- **Disposal:** Ensuring the robot can deposit collected waste into designated trash cans.
- **Operational Efficiency:** Completing cleaning cycles within the allocated time and adhering to budget constraints.

The main challenges of this project are twofold; on the software side, developing reliable navigation and obstacle avoidance algorithms to ensure the robot can efficiently manoeuvre park terrains without encountering water or obstacles (people / chairs). On the hardware side, designing a robust waste collection and disposal system that can effectively gather and deposit multiple pieces of dog waste within the given constraints requires innovative thinking and thorough testing.

To serve as a proof of concept for stakeholders, this first prototype's trial will take place on a 122cm x 122cm park map with the following ground colors to represent the different terrain the robot will encounter: green representing safe grass to traverse, blue symbolising bodies of water to avoid, yellow to mark the trash collection area, and a red grid which can be used to aid navigation. The objects that the robot will encounter will be simulated by colored cube markers (purple: people, green: seats, orange/yellow: dog waste). (Appendix Figure B.1)

## B. Requirements

In order to meet the client's expectations, our design solution must fulfil the following requirements which we distilled from the client needs:

**R1. Navigation and Obstacle Avoidance**

- **R1.1:** The robot shall autonomously navigate all accessible areas of the park
- **R1.2:** The robot shall detect and avoid obstacles such as water bodies, benches, and people during navigation to ensure uninterrupted movement and safety.
- **R1.3**: The robot shall have the ability to complete precise turns, driven by a navigation algorithm.
- **R1.4:** The robot shall have the ability to travel along precise and consistent straight paths as required by a navigation algorithm.

**R2. Waste Collection Capacity**

- **R2.1:** The robot shall be capable of locating and picking up waste cubes.
- **R2.2:** The robot shall securely store a minimum of six waste cubes.
- **R2.3:** The robot shall transport collected waste cubes to the designated trash collection area for disposal.

**R3. Waste Deposition Frequency**

- **R3.1:** The robot shall deposit collected waste cubes frequently enough to prevent exceeding its maximum carrying capacity, ensuring efficient operation without overloading.

**R4. Task Completion and Return**

- **R4.1:** Upon completing its waste collection and deposition tasks, the robot shall autonomously return to its designated starting point.

**R5. Unreachable Waste Detection**

- **R5.1:** The robot shall avoid picking up waste cubes that are unreachable (e.g. located in water or other inaccessible areas).

**R6. Physical Dimension and Footprint**

- **R6.1:** The robot's hardware footprint shall be sufficiently narrow to allow it to traverse the park bridge of 17.8 cm in width without contacting the surrounding water.

## C. Constraints

Furthermore, the design space of our prototype will be restricted by the following constraints:

**C1. Time Constraints**

- **C1.1:** The robot shall complete its task of navigating through the park and collecting the waste cubes within a maximum of 3 minutes.
- **C1.2:** The robot must be fully designed, built, and ready for operation within a period of 5 weeks from the project commencement date.

**C2. Hardware Limitations**

- **C2.1:** The robot shall utilize no more than 4 sensors, adhering to the limitations imposed by the BrickPi system.
- **C2.2:** The robot shall be equipped with a maximum of 4 motors, in compliance with the BrickPi hardware capabilities.

**C3. Resource Allocation**

- **C3.1:** The project shall respect a time budget of an average of 9 hours per team member per week, encompassing all project-related activities.

**C4. Material Restrictions**

- **C4.1:** The robot's construction shall be limited to components provided in the BrickPi kit and approved common household materials.
- **C4.2:** The use of any additional materials or specialised components outside of the approved list is prohibited unless prior approval is obtained.

## D. Specifications

In order to meet the requirements while respecting the constraints, the solution will be defined by the following specifications:

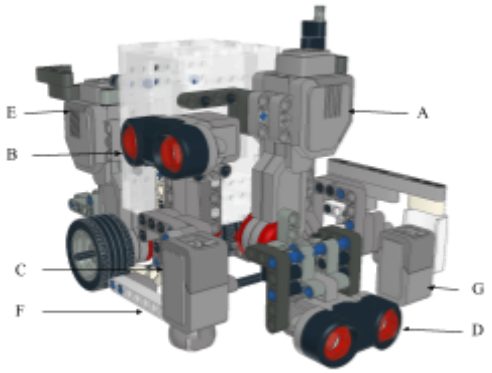



Figure 3. Specification Mapping

| Map | Component | Specifications |
|---|---|---|
| **A** | LMF | -Ability to precisely rotate the USR **180 degrees** up and down, in response to nav algorithm |
| **B** | USR | -Ability to detect side wall with range of **255 cm**, enabling wall following |
| **C** | CSR | -Scan and classify **2.54 cm²** cubes into **3 categories**: people (purple), seats (green), waste (yellow, orange)<br>-Scan and classify the ground into **4 categories**: grid (red), water (blue), grass (green), trash (yellow) |
| **D** | USF | -When lowered, has the ability to detect and precisely locate **2.54 cm²** cubes<br>-When raised, has the ability to detect the forward wall with a range of **255 cm** |
| **E** | LML, LMR | -Precisely drives a sweep of the **122cm x 122cm** map, turning and driving in line<br>-Has the ability to respond to sensor data, based on the navigation algorithm |
| **F** | Cube Cage and Chassis | -Carrying cage of dimension **9 cm x 6.2 cm**, capable of carrying at least **6 2.54 cm²** cubes<br>-Total footprint of **428.75 cm²** & wheel base of **17.5cm** able to fit on the bridge of width **17.8m** (See Appendix Table B.1. for the total robot dimensions)<br>-Ability to deposit cubes on the trash are of dimension **17.3cm x 17.3cm** |
| **G** | CSL | -Scan and classify the ground into **4 categories**: grid (red), water (blue), grass (green), trash (yellow) |

Table 4: Robot Measures and Specifications

## E. Assumptions and Additional Considerations

Through conversations with our designated Senior Engineer and with the client, we have derived and confirmed the following assumptions regarding the project:

- **A1.** All sources of water are convex in shape.
- **A2.** Water sources cannot touch the wall and are located within the red grid.
- **A3.** Cubes cannot touch the wall and are located within the red grid.
- **A4.** Physical walls will provide a barrier to the park.
- **A5.** Two cubes of different colour cannot be within a certain distance of each other (2 inches).
- **A6.** Once waste is deposited in the given location, it can be removed.
- **A7.** Distance between the two bodies of water is at least 7 inches.
- **A8.** There will be approximately 8-10 cubes outside the water and 2-3 inside.
- **A9.** All obstacles and waste cubes are stationary during the robot's task.

# III. Design

## A. System Design Selection Process

This section details the methodology and criteria used to select the initial system design basis, at the start of the design process. It will provide insight into our decision-making process, showcasing how different design alternatives were evaluated and chosen based on specific requirements and constraints to best fit the client's needs.

### Design Philosophy

Our design philosophy is deeply influenced by the experience and strengths of our team, which comprises primarily software engineering students. Recognizing this, we prioritised the development of robust navigation and pathfinding algorithms as the cornerstone of the robot's functionality. Simultaneously, we emphasised minimal hardware complexity to create a compact and efficient robot, allowing us to put our main focus on what we believe to be the crux of the challenge: the complex software algorithms which drive our design ideation process.

The first stages of our design process were steered by an initial extensive brainstorming meeting conducted on Oct. 31st. During this meeting we discussed the following main three build ideas to serve as a basis for our design process:

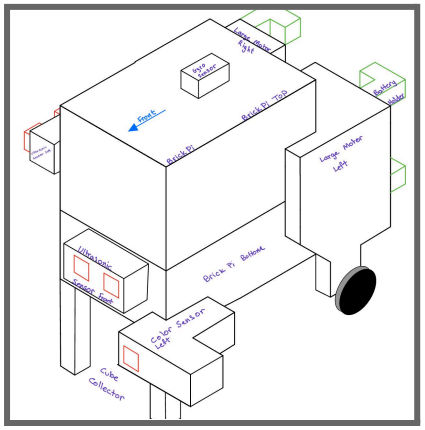



Figure 4. Build 1 Labeled Isometric Schematic

### Build 1: Dual Brick Robot (Not Chosen)

The Dual Brick Robot design features a two-brick setup designed for efficient navigation and object interaction (for full specifications, see Appendix Doc. C1). The robot utilizes a cage-like base supported by ball bearings and wheels for movement, driven by two motors. Its sensor system includes a forward-facing ultrasonic sensor for obstacle detection, a right-facing ultrasonic sensor for wall-following, a front-left color sensor for water mapping, a forward-facing color sensor for block identification, and a centrally positioned gyro sensor for orientation and navigation precision. The robot's navigation strategy involves a counterclockwise field sweep, water mapping to avoid obstacles, wall-following for boundary alignment, cube detection, and block color categorization to optimize its task performance. Two bricks are used to increase our sensor capacity.

| Pros | Cons |
|---|---|
| **Efficient Navigation:** Well-defined navigation path with specific sensors for water mapping and wall following. | **Heavy Build:** The setup is heavy due to multiple components, potentially affecting mobility and battery life. |
| **Enhanced Sensor Feedback:** Multiple sensors provide redundant data, reducing the chance of errors and increasing reliability. | **Mechanical Complexity:** More components increase the risk of mechanical issues, requiring careful construction and testing. |
| **Comprehensive Field Coverage:** The planned navigation ensures a complete sweep of the field, systematically covering all areas. | **Slow Brick Communication:** Communication between the two bricks may be slow, potentially impacting sensor responsiveness and control timing. |

Table 5. Build 1 Pros v.s Cons

### Build 2: Arm-based Robot (Not Chosen)

The Arm-based Robot design employs a two-axis arm equipped with a color sensor to perform targeted cube collection within a grid-based navigation system (for full specifications, see Appendix Doc C.2). The arm extends over individual squares to detect and analyze cubes, allowing for efficient scanning without driving over every area. The robot's four motors include two for

movement and two for arm control along the X and Y axes. Its sensor suite comprises an ultrasonic sensor for obstacle and cube detection, a front-left color sensor for water hazard avoidance, an arm-mounted color sensor for cube identification, and a gyro sensor for directional orientation. Using a square-by-square progression, the robot systematically covers the field, aligning with grid lines for precise positioning. It scans each square using the ultrasonic sensor's 45-degree sweep, identifies cube colors via the arm-mounted sensor, and avoids water hazards using the front-left color sensor. This method ensures thorough coverage and efficient collection of desired cubes while maintaining a structured, grid-based approach.

| Pros | Cons |
|---|---|
| **Comprehensive Coverage:** The grid-based approach ensures a complete sweep of the field, reducing the likelihood of missed areas or objects | **Complex Arm Mechanics:** The dual-axis arm for probing is mechanically challenging to build and may be prone to errors, particularly if alignment or calibration is off. |
| **Proactive Obstacle Detection:** Each square is assessed before entry, minimizing the risk of collision with obstacles. | **Slow Progression:** Moving square by square with a probing and scanning routine is time-consuming, making this approach potentially slower than previous builds. |
| **Structured Navigation:** Using a standard grid makes navigation predictable and easy to monitor, especially with the gyro sensor providing directional accuracy. | |

Table 6. Build 2 Pros v.s Cons

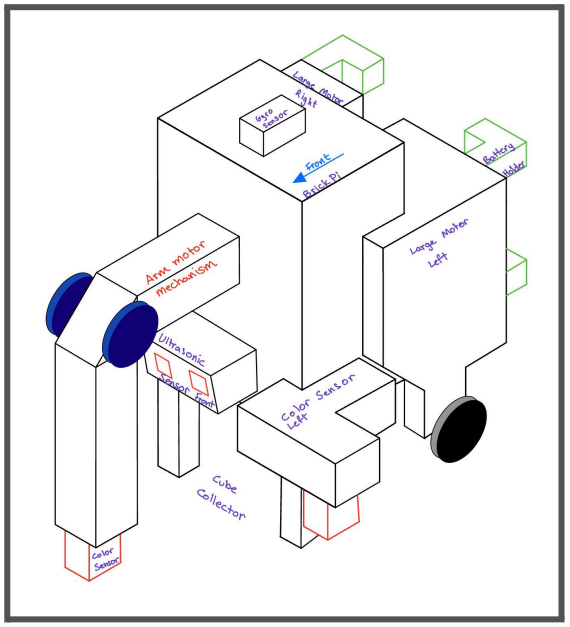



Figure 5. Build 2 Labeled Isometric Schematic

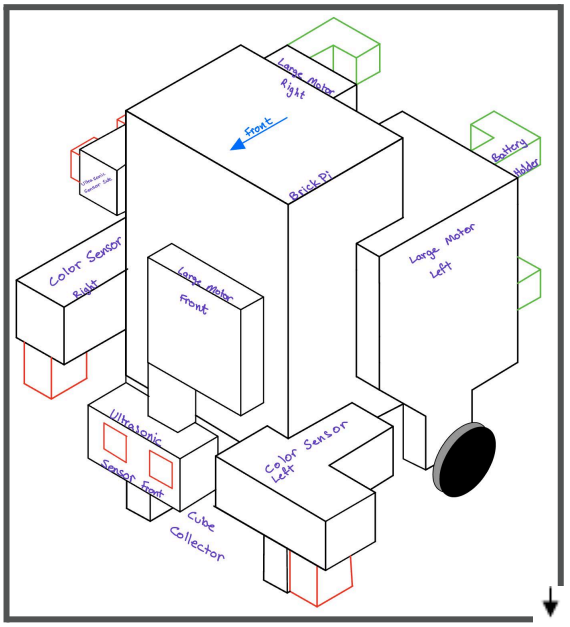



Figure 6. Build 3 Labeled Isometric Schematic

Build 3: Compact Spiral Navigation Robot (Chosen)

The Compact Spiral Navigation Robot provides a lightweight and simplified basis for our design, featuring a single brick to reduce complexity and weight while eliminating the need for inter-brick communication (full specification in Section 3.B-D). Its primary innovation is a motorized, rotating multi-purpose ultrasonic (US) sensor that adjusts vertically to detect cubes and walls depending on the mode, replacing the need for a gyro sensor. The robot uses two motors for movement and one motor for the US sensor's rotation.

The sensor system includes a rotating front-facing US sensor for multi-height detection, a right-facing US sensor for wall-following navigation, a downward-facing right color sensor to detect water and scan cubes, and a downward-facing left color sensor for ground detection and trash area identification. The navigation strategy emphasizes simplicity and efficiency, focusing on a primary spiral path across the field. The robot avoids water directly using the both color sensors to adjust motor speed to turn and follows walls using the rotating US sensor, and detects cubes with the right CS sensor. This system allows for rapid navigation, obstacle avoidance, and cube collection while ensuring a lightweight, streamlined build.

| Pros | Cons |
|---|---|
| **Simplified Build:** Single brick design and fewer sensors reduce complexity and mechanical risks. Lightweight and Compact: Easier to maneuver and less prone to physical strain on components. | **Potential US Sensor Errors:** Moving the US sensor up and down might introduce inconsistencies, particularly if it's positioned incorrectly, leading to mixed signals (e.g., partially detecting both wall and cube). |
| **Efficient Navigation:** Allows full field sweep without needing to map water areas extensively, focusing on obstacle avoidance and main path adherence. | **Reduced Sensor Redundancy:** Fewer sensors increase reliance on the remaining US sensors, which may be more error-prone. |

Table 7. Build 3 Pros v.s Cons

Conclusion: Why Build 3 is the Ideal Candidate

We chose Build 3 as the basis for iterative refinement because it strikes the perfect balance between simplicity, efficiency, and functionality. Its lightweight and compact design reduces mechanical and communication complexities, making it easier to construct, test and refine. By utilizing a single brick and a strategic placement of sensors, this design focuses on maneuverable navigation and obstacle avoidance without overcomplicating the system.

Moreover, its emphasis on a simplified approach to water detection and cube collection aligns well with our design philosophy of prioritizing robust algorithms over hardware intricacy. While it sacrifices some redundancy, the minimized risk of mechanical failure and its adaptability to iterative improvements make Build 3 the most practical and scalable option to meet the client's needs.

## B. System Description Overview

Hardware Sensor and Motor Nomenclature

| Acronym | Name | Description | Port |
|---|---|---|---|
| LML | Large Motor Left | Large motor connected to a tire, located in the back left of the robot which drives its movement | MC |
| LMR | Large Motor Right | Large motor connected to a tire, located in the back right of the robot which drives its movement | MB |
| LMF | Large Motor Front | Large motor connected to a USF, located in the front center of the robot which drives the rotation of USF | MA |
| CSL | Color Sensor Left | Color sensor located x cm above ground level on the front left of the robot, responsible for detecting the ground type. | S3 |
| CSR | Color Sensor Right | Color sensor located x cm above ground level on the front right of the robot, responsible for detecting the ground type as well as blocks. | S2 |
| USF | UltraSonic Front | Ultrasonic sensor, located in the front centre of the robot mounted to the LMF, responsible for located blocks when lowered and for | S4 |

| | | | |
|---|---|---|---|
| | | detecting the forward wall when up. | |
| USR | UltraSonic Right | Ultrasonic sensor, located on the top right side of the robot, responsible for polling the outside wall, ensuring a parallel movement path. | S1 |

Table 8. Hardware Nomenclature and Port Mapping

System Hierarchy Model

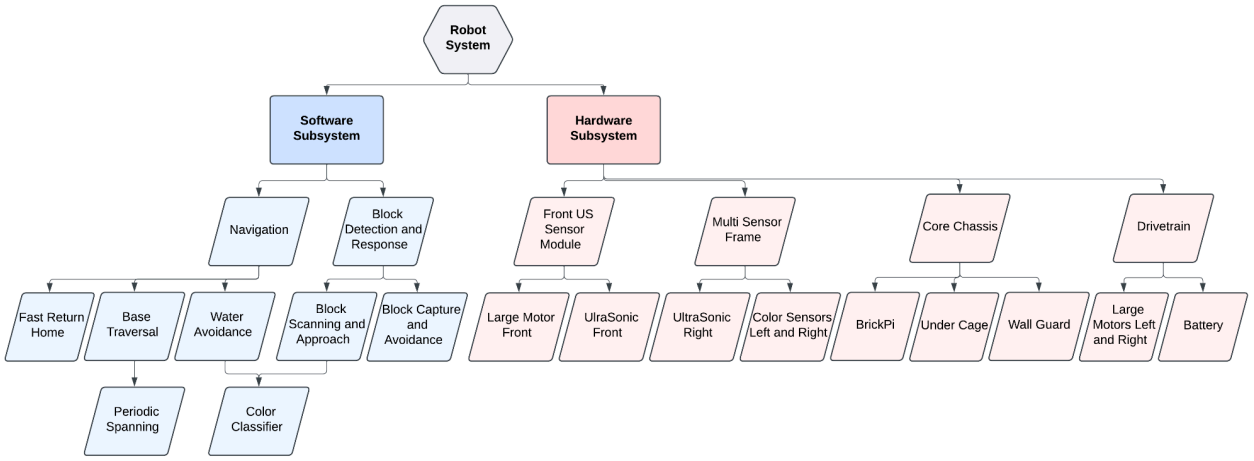



Figure 7. Robot System Hierarchy Model

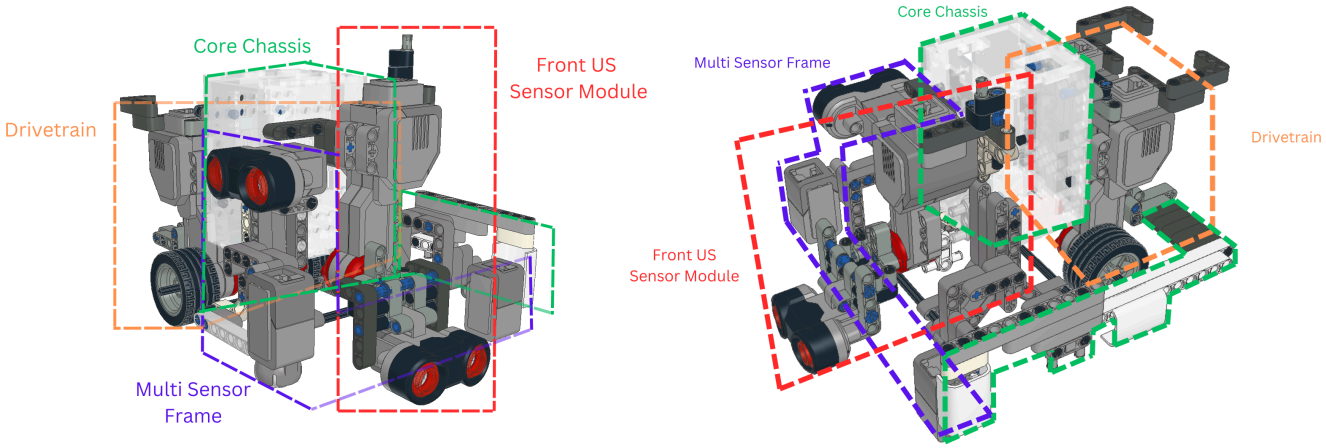



Figure 8. Hardware Component Breakdown

## C. Hardware Subsystems and Iterative Evolution

The analysis and description of the hardware will be done through the iterative evolution of each of the 4 subsystems detailed above in Figure 7 and 8.

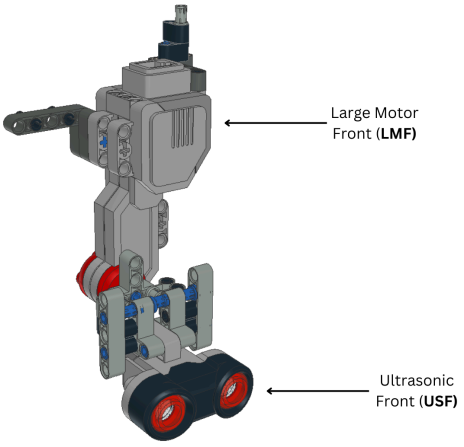



Figure 9. Front US Sensor Module

**Front US Sensor Module**

Overview:

The Front Ultrasonic Sensor Module integrates a motorized ultrasonic sensor capable of vertical rotation between a high and low position. Its tasks include detecting cubes, identifying walls, and aiding in robot

orientation by measuring distance to the wall in front of the robot. (See Appendix Image C.7.)

Specifications:
- Rotation Range: 180 degrees
- Rotation Accuracy: Less than 5 degrees of displacement over 10 cycles
- Rotation Positions & Height: Sensor should have an up position with a field of view above 2.5cm and a bottom position at a maximum of 2 cm from the ground
- Cable Management: Cable shouldn't obstruct or have any impact on the rotation of the sensor.

| Iteration | Description | Components | Changes |
|---|---|---|---|
| 1 | Introduced a motorized US sensor for dual-purpose object detection and wall alignment. | -Rotating Ultrasonic Sensor (USF)<br>-Large Motor Front (LMF) | **Changes:** N/A<br><br>**Issues;**<br>Sensor was too close to the motor which meant the cable obstructed free rotation of USF. |
| Final | The final iteration ensured a stable and efficient system for navigation and object detection. It combined a stable cable length through holders and the rotation was stable through cycles. Finally the up height was at 7cm and the low position was at 1.2cm. | -Rotating Ultrasonic Sensor (USF)<br>-Large Motor Front (LMF) | **Changes:**<br>-Added distance between motor and US sensor with a longer attachment.<br>-Bent the cable and secured it using the new attachment atop the sensor so it doesn't interfere in rotation (See Appendix Image C.9-C.10)<br>-Secured the cable on top of the motor with a flexible attachment so the cable length between the top of the motor and the sensor is always the same and ensures consistent unobstructed movement. (See Appendix Image C.8-C.11)<br><br>**Issues:** No issues |

Table 9. Front US Sensor Module Design Iterations

**Multi Sensor Frame**

Overview:
The Multi Sensor Frame is mounted at the front of the robot and houses all the static sensors used for navigation, cube detection, water avoidance, and wall following. Additionally, it incorporates rolling metal balls to aid the robot's movement and navigation and supports the weight in front. (See Appendix Image C.6.)

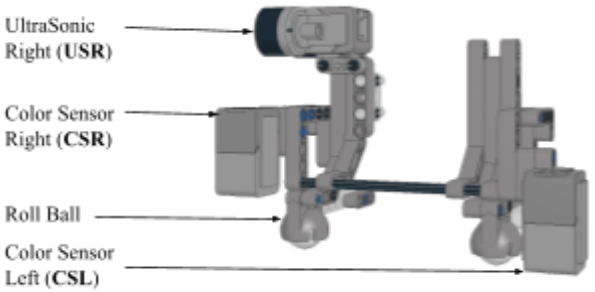



Figure 10. Multi Sensor Frame Module

Specifications:
- Color sensor height: One of the sensors needs to be higher than 2.5cm to detect cube color
- Color sensor width: The sensors should be wider than the wheel base
- Color sensor and ball distance: Sensors should be at least 1 cm from the metal balls to avoid water
- Side US Sensor Placement: Sensor should be facing right at a 90 degree angle and be higher than 2.5 cm high to avoid detecting cubes.
- Middle of the Frame Gap: The middle gap should be higher than 2.5 cm to let cubes go under.
- The two metal balls should be at least 2.5 cube widths apart

| Iteration | Description | Components | Changes & Issues |
|---|---|---|---|
| 1 | Initial design had 2 color sensors for water avoidance, with the right one placed 2.7cm high to also detect cubes. The US sensor was placed at a 90 degree angle facing right. The frame hinged on two metal balls placed 1cm in the interior diagonal of their respective color sensor. | -Two color sensors (CSL, CSR)<br>-Ultrasonic Sensor (USS)<br>-Rolling Metal Balls | **Changes:** N/A<br><br>**Issues**<br>-Color sensors were too narrow which didn't leave the robot enough space to avoid water properly.<br>-Metal balls were placed too close together, reducing stability only allowing two cubes in if they were perfectly aligned. |
| 2 | Same design as iteration 1 with a larger sensor and metal ball width to have more cube pickup leeway and greater water avoidance. The US sensor is still positioned at 9.5cm high, the color sensors are 15cm apart and the metal balls are 9.5cm apart. The lowest point of the frame in the gap stands at 3cm. (See Appendix Image C.3.) | -Two color sensors (CSL, CSR)<br>-Ultrasonic Sensor (USS)<br>-Rolling Metal Balls | **Changes**<br>-Increased the gap between color sensors by 2 cm on either side to extend coverage beyond the wheelbase for better water avoidance.<br>-Increased the gap between metal balls by 1 cm on each side to improve robot balance and increase space to gather cubes.<br>**Issues**<br>-The left side color sensor was so far out that we'd adjust too much out when seeing water which caused the reversal to the mean path to clip water from the back wheels. |
| Final | Refined final positions of both color sensors while keeping the chassis the exact same. Distance between color sensors is now 14cm. | -Two color sensors (CSL, CSR)<br>-Ultrasonic Sensor (USS)<br>-Metal Balls | **Changes**<br>-Brought both color sensors in by 0.5cm<br>**Issues**<br>-No more issues in this subsystem. |

Table 10. Multi Sensor Frame Design Iterations

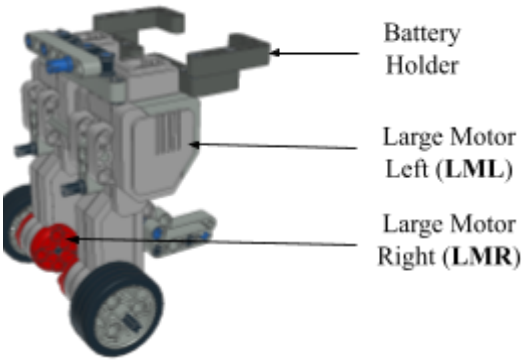



Figure 11. Drivetrain Module

**Drivetrain**

Overview

The Drive Chassis module provides the core movement functionality for the robot, utilizing dual motorized wheels and structural stability through a centralized battery placement. It ensures balanced, efficient navigation and supports modular connectivity with the robot's subsystems. (See Appendix Image C.4.)

Specifications:

- Width: Less than 17.8 cm to fit on the bridge.
- Motors: Dual large motors for independent left (LML) and right (LMR) wheel control.
- Battery Holder: Should allow no battery movement during regular operation and easy change of battery.

- Motor Orientation: It should maximize the weight over the wheels to increase friction and should allow for maximal space below the robot for cube gathering

| Iteration | Description | Components | Changes |
|---|---|---|---|
| 1 | Initial design of the drive chassis focused on dual-motorized wheels with standing orientation motors with the battery to the side of the robot (See Appendix C.17.) for easy retrieval. The outer part of the wheels were 12.5cm apart which fit on the bridge. | -Large Motor Left (LML) and Large Motor Right (LMR): Drive motors for navigation<br>Side Battery Holder held the battery at the side of the robot and enabled easy access | **Changes:** N/A<br><br>**Issues:**<br>-The Battery Holder was flimsy and the battery would wiggle a little during driving<br>-The weight was unequally distributed with the battery at the side of the robot |
| Final | The final iteration provided a robust and balanced drive train. Improvements include a better weight distribution with the battery's weight centered above the wheels for consistent performance. The inner wheel base was 8.2cm in width and the outer width still stood at 12.5cm. | Large Motor Left (LML) and Large Motor Right (LMR): Drive motors for navigation<br><br>Back Mounted Battery Holder holds the battery behind the drivetrain of the motors. | **Changes:**<br>Created a holder at the back of the drivetrain and passed the cable between the motors to connect to the battery ensuring a centralised weight distribution without neglecting easy access<br><br>**Issues:** No issues |

Table 10. Drivetrain Module Design Iterations

**Core Chassis**

Overview

The core chassis module provides the core frame on which the other modules are attached and includes the BrickPi, the under-mounted cage and the wall guard. It is a static part which supports the attachment of the other submodules and protects the robot. (See Appendix Image C.5.)

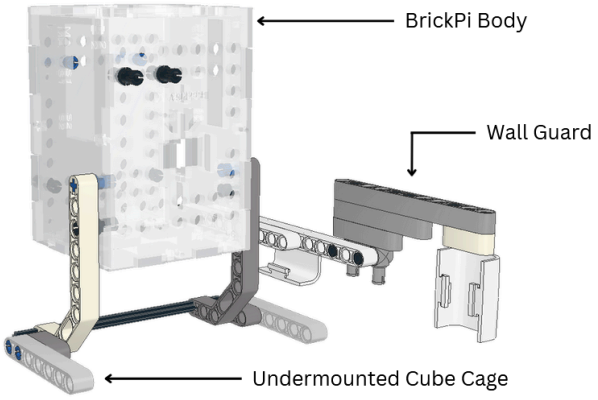



Figure 12. Core Chassis Module

Specifications:

- Cage Size: The cage should be able to hold 6 cubes in total.
- Sturdiness: The wall guard should allow the robot to lean on it without bending or breaking
- Smoothness: The guard should not get caught in the wall when the robot pushes against it.
- BrickPi Height: The Pi should be at least 2.5cm high so as to not touch the cubes underneath.

| Iteration | Description | Components | Changes |
|---|---|---|---|
| 1 | The initial design was extremely simple consisting only of the BrickPi at a 5cm height and the under-mounted | -BrickPi<br>-Under-Mounted Cage | **Changes:** N/A<br><br>**Issues:** |

| | | | |
|---|---|---|---|
| | cage which had a 5 cm inner width. | | -Although 5cm was enough to hold 2 cubes side by side, it allowed for very little leeway which meant that when we were picking up cubes, we'd need perfect alignment for the cube to go into the cage. |
| 2 | Same design as the previous design but this one included a larger cage of inner width 6.2cm. The total area of the cage was 46.5cm$^2$ | -BrickPi<br>-Under-Mounted Cage | **Changes:**<br>-We realized that although 6 cubes could fit, we needed leeway to account for the variability in the robot's environment and extended the cage by 1.2cm which allowed for extra cubes, effectively making the cage able to hold 8 cubes.<br><br>**Issues:**<br>When implementing the "Fast Return Home" method, our robot would lean onto our color sensor getting stuck. |
| Final | The final iteration kept the same cage design but now implemented a new addition to the subsystem. The wall guard was meant as a structural element against which the robot could lean when reverting home. | -BrickPi<br>-Under-Mounted Cage<br>-Wall Guard | **Changes:**<br>Created a wall guard that attaches to the left motor of the drivetrain and the left side of the multi-sensor frame. It implements a protection around the sensor and two major points of attachment for maximum stability when the robot leans against it.<br><br>**Issues:** No issues |

Table 11. Core Chassis Module Design Iterations

**Final Assembly & Modularity**

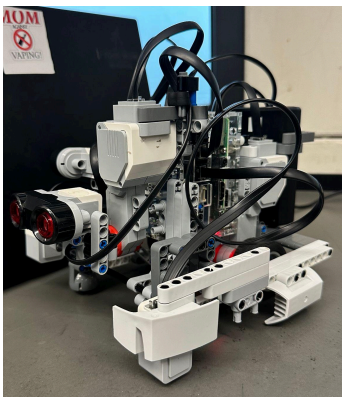


Figure 13. Picture of Final Robot Assembly

When building the initial design of this robot (See Appendix C.17.), we right away set out to make a modular robot. Each subsystem attached to the others without any additional parts, and each subsystem could go through its own iterative process without hindering the others. At first, we had 4 separate detachable parts. When assembled, we didn't meet any integration problems and we didn't have any subsystem integration changes. All the iterations were constrained to their specific subsystems and were aligned to better meet the specifications of that subsystem. The different subsystems Front US sensor subsystem interacted with the Sensor Frame subsystem in cube detection and color identification and the sensor frame interacted with the Drivetrain for navigation through the color and US sensors mounted on it. The Front US sensor also interacted with the Drivetrain when detecting cubes. Everything was managed through the BrickPi in the Core Chassis Subsystem. For our final design, we had 5 separate parts as the wall guard attaches to the outside and stands alone when the robot is deconstructed although it is part of the core chassis subsystem. To assemble the robot, each of the 5 parts need to be clipped onto the main chassis with connectors in no particular order (See Appendix C13-14-15) - except for the wall guard which needs to be attached last (See Appendix C.16). This allowed us to be able to change the BrickPi or Motors in minutes when identifying errors and providing a faster iterative design. See Appendix C.12 for our final, deconstructed robot, into its 4 subsystems in 5 parts.

## D. Software Subsystems Iterative Evolution

### Software Architecture Overview

The team employed modular programming techniques to organize the software architecture. The robot's software was divided into two primary modules: *Robot Navigation* and *Block Detection and Response*. Each module was composed of several algorithms which were generated by creating and applying functions in Python. The Robot Navigation module was responsible for the robots traversal of the course and included the robot's Base Traversal algorithm, the Water Avoidance algorithm, and the Base Traversal with Spanning algorithm. The Block Detection and Response module was responsible for detecting cubes using the front ultrasonic sensor, safely approaching and scanning them, and either capturing or avoiding them based on the cube colour. The Block Detection and Response module employed the Block Detection algorithm and the Classification, Capture, and Avoidance algorithm. A Color Classification algorithm was employed by both modules. The two modules were integrated using the Integration algorithm. See Appendix Table D.3 for a description of the responsibilities and utilized algorithms for each module. The following section describes the algorithms making up both modules as well as the ultimate integration of the two modules into the finalized software subsystem.

### Algorithms Common to Both Modules

**Color Classification Algorithm**

To accurately classify the various colors the robot will encounter, we developed two distinct color classification models, one tailored to each sensor. This process involved collecting approximately 20-30 data points for each color, separately for each sensor. This data is given in Appendix Tables C.1 and C.2. From this data, we calculated the normalized center values for each color, which serve as reference points. These centers allow us to compute the Euclidean distance in color space from a new data point, enabling precise and consistent color identification. These models will serve as the basis for both ground and block detection, enabling the robot to take action based on the data it receives from its environment. A separate model was developed for each of the color sensors due to the sensors' differences in heights to the ground (CSR high and CSL low). Figure 14 provides a visualization of the colour space used to calculate the Euclidean distance to the colour centers. See Appendix Figure C.1 and C.2 for further visualizations of the colour space. See Appendix Figure C.3 for a flow chart visualisation of this algorithm.

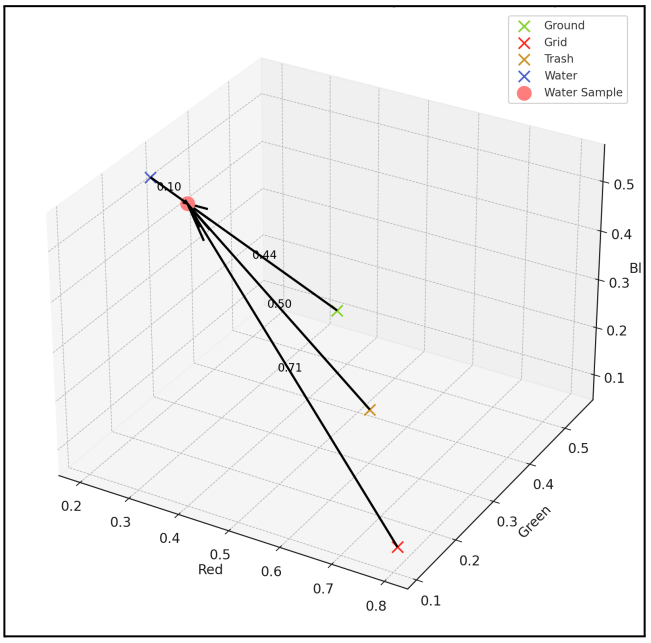



Figure 14. Classification of CSL Water Sample Point in Normalized RGB Colorspace

## Block Detection and Response

### Block Detection and Approach Algorithm

The goal of the Block Detection and Approach Algorithm was to detect and approach cubes to capture or avoid them. To achieve this, the USF is used to detect when a cube is within 15 cm of the robot. When an object is detected in this range, the USF is rotated upwards above the height of a cube using the LMF. This is done to determine if the detected object is a cube or the wall. If the difference between the polled distance when the USF is in the upward position compared to when it is in the downward position is above a set threshold, the object is classified as a cube. If it is below the threshold it is classified as the wall. If the detected object is classified as a cube, the robot begins to localize the position of the cube. To do this, the robot begins rotating to the left until the polled distance significantly increases, indicating the USF has passed the left edge of the cube. The robot then rotates right until once again the polled distance drastically increases, indicating the USF has just passed the right edge of the cube. The middle point in between the left and right edge of the cube is its center point. Once the exact position of the cube has been localized, the robot begins approaching until it is within 3 cm of the cube. Appendix Figure C.3 provides a visualization for the algorithm in the form of a flow chart. The utilized sensors and motors are detailed in Appendix Table C.4.

Iterations**:**

The Block Detection and Approach algorithm underwent several iterations. A summary of these iterations is given in Table 12. See Table C.5 in the Appendix for a more detailed description of the iterations.

| Iteration Number | Description |
|---|---|
| V1.0 | The USF is used to detect a block, and rotate to determine its exact position relative to the robot. |
| V1.1 | The USF is used to approach the block until it is close enough to rotate. |
| V1.2 | After first detecting an object with the USF, the LMF rotates the USF to determine if the object is a cube. |

Table 12: Iterations of Cube Detection and Approach Algorithm

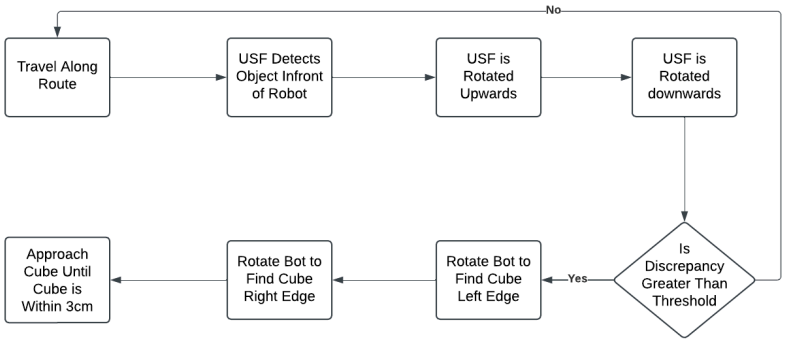



Figure 15: Flow Chart Visualization of Cube Detection and Approach Algorithm

### Classification, Capture and Avoidance Algorithm

After the robot has approached the cube, the robot begins rotating left to position the cube directly below the CSR. The CSR then polls the colour of the cube below it and uses the Colour Classification algorithm to determine the colour. If the cube is classified as a poop colour (Yellow or Orange), the capture algorithm is triggered. To capture the cube, the robot will raise the USF using the LMF, rotate slightly to the right, drive forward, then lower the USF causing the cube to be collected in the undermounted cage. If the cube is classified as an obstacle colour (Green or Purple),

the avoidance algorithm is triggered. To avoid a cube, the robot will drive backwards slightly. Figure 16 provides a visualization for the algorithm in the form of a flow chart.

Iterations

The Classification, Capture and Avoidance algorithm underwent several iterations detailed in Table 13. See Appendix Table C.7 for a more detailed description of the iterations.

| Iteration Number | Description |
|---|---|
| V2.0 | The robot can turn and move forward slightly until the CSR is directly above the cube it is trying to classify. |
| V2.1 | The robot's approach_cube() function was changed to allow it to reach cubes more accurately and consistently. |
| V2.2 | The robot can accurately reach a cube and determine its color and decide whether it should be avoided or picked up. |
| V2.3 | The robot can successfully pick up cubes it determines to be poop. |
| V2.4 | The robot successfully backs up slightly to avoid cubes. |

Table 13: Iterations of Cube Classification, Capture, Avoidance, Algorithm

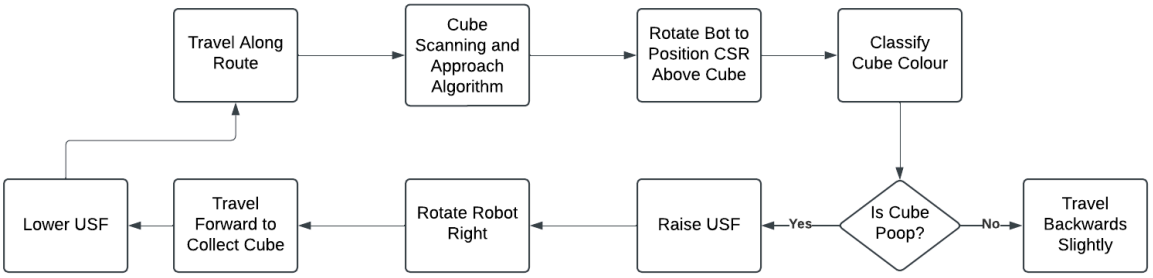



Figure 16. Cube Classification, Capture, and Avoidance

## Robot Navigation

**Base Traversal Algorithm**

The initial traversal algorithm aimed to traverse the majority of the map by traveling in iterative spirals. The robot would begin in the 'Start' corner and travel along the outside walls. The robot utilized the USS to poll the adjacent wall to maintain a target distance from the side wall. If the robot drifted further away from the side wall than the target side distance, the left wheel speed was increased. When the robot drifted closer to the wall than the side distance target, the robot's right wheel speed was increased. Simultaneously, the robot would poll the distance to the wall in front of it using the USF. When the distance between the USF and the wall dropped below a set distance threshold, the robot would make a 90° turn to the left. The robot would then once again follow along the adjacent wall while maintaining a set side wall target distance. After the robot completed three left turns, the front wall distance threshold was increased. After the next left turn, the side distance target was also increased. This process then repeated, allowing the robot to complete multiple spiral iterations, progressively moving inwards to cover most of the map, successfully implementing wall based self-location. This traversal algorithm ignored obstacles and water. A visualization of this traversal algorithm is provided in Figure 17 and Figure 18. A legend for Figure 17 is provided in

Appendix Table C.8 (See Appendix). A description of the sensors and motors used in the Base Traversal algorithm is provided in Appendix Table C.9. See Appendix Figure C.4 for a flowchart visualization of the algorithm.

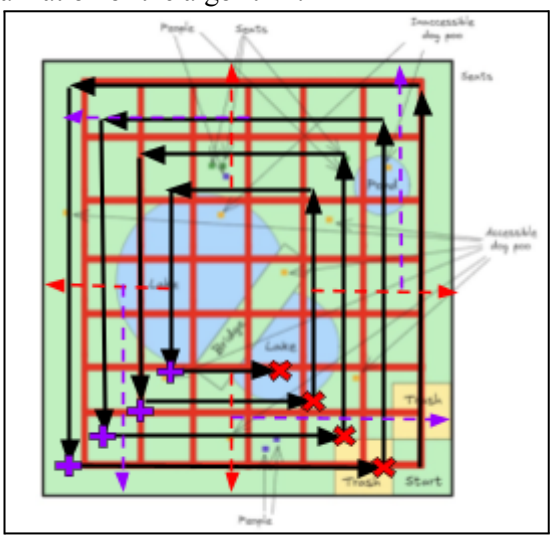



Figure 17. Base Traversal Algorithm Visualization

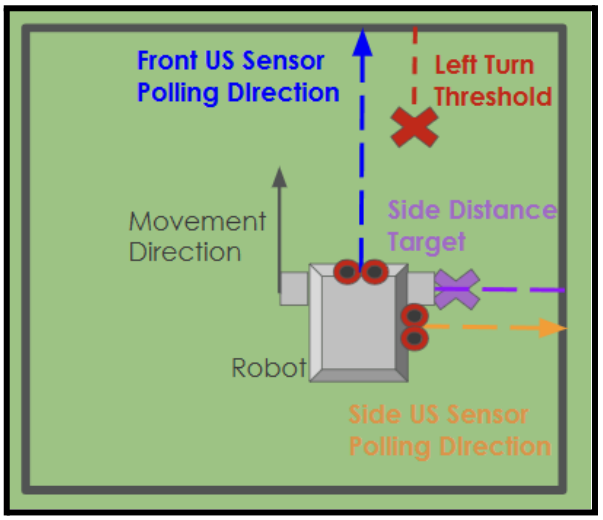



Figure 18. Wall-based self-location visualisation

Iterations

The Base Traversal algorithm underwent two key iterations. These iterations are detailed in Table 14. See Appendix Table C.10 for a more detailed description of the iterations.

| Iteration Number | Description |
|---|---|
| V1.0 | Robot used only USF to detect the forward wall and turn left. Robot could not maintain parallel movement to the side wall. |
| V1.1 | Robot used both USF to detect forward wall and USS to detect side wall to maintain parallel movement |

Table 14: Iterations of Base Traversal Algorithm

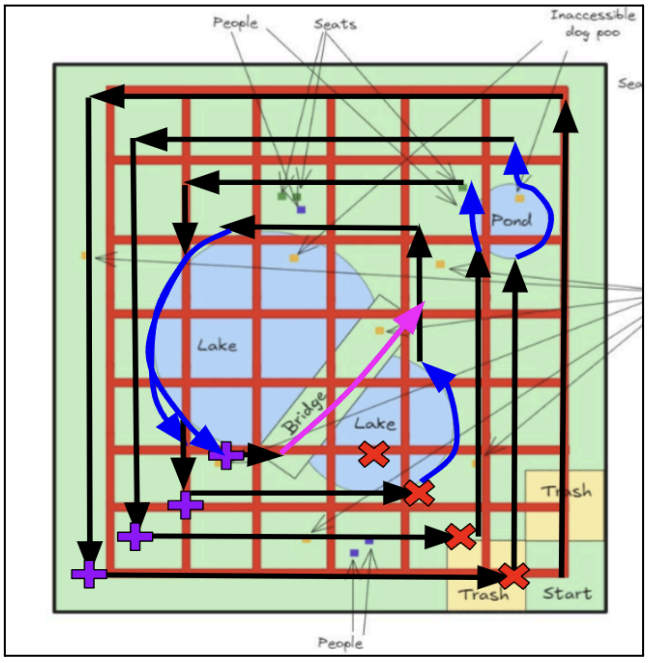



Figure 19. Water Avoidance Visualization

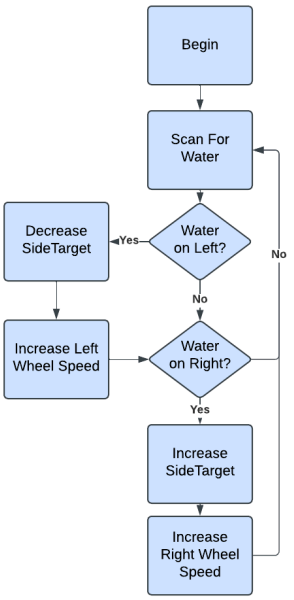



Figure 20. Flowchart Water Avoidance

**Water Avoidance Algorithm**

The next adaptation to our traversal algorithm was implementing a water avoidance algorithm. To achieve water avoidance, two color sensors were positioned on the front left and front right corners of the robot facing downwards to continuously poll the ground. The readings from the colour sensor's were passed into the Colour Classification Algorithm. The robot would begin by following the spiral path in accordance with the base traversal algorithm. However, if the robot detected water with either of its colour sensors it would increase the speed of the appropriate wheel to evade the water. For example, if the colour sensor in the front left corner detected water, the speed of the left wheel would increase, allowing the robot to bend right around the water. Once the water had been successfully evaded, the robot would return to following the basic spiral traversal algorithm. This algorithm would allow the robot to navigate the entirety of the accessible map. Obstacle cubes were still ignored in this algorithm. Figure 19 provides a visualization of the Water Avoidance algorithm path (See Appendix Table C.11 for a legend). Figure 20 provides a visualization of the algorithm in the form of a flowchart. A description of the sensors and motors used in the Water Avoidance algorithm is provided in Appendix Table C.12.

Iterations

The Water Avoidance algorithm underwent several key iterations. These iterations are detailed in Table 15. See Appendix Table C.13 for a more detailed description of the iterations.

| **Iteration Number** | **Description** |
|---|---|
| V2.0 | The robot uses the color sensors to only poll the ground and display to the user the detected color from each sensor |
| V2.1 | The robot uses CSL to avoid water detected on the left of the robot. |
| V2.2 | The robot uses both sensors to avoid both the water on the left and the right side of the robot. Robot cannot avoid water that is directly in front of it. |

Table 15: Iterations of Water Avoidance Algorithm

**Base Traversal with Spanning Algorithm**

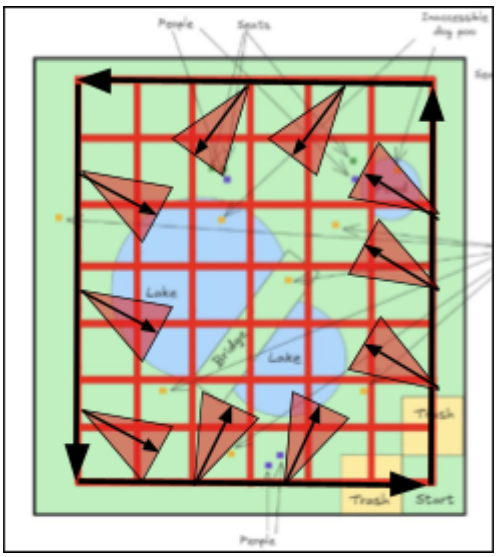



Figure 21. Base Traversal with Spanning

After rigorous testing of the water avoidance algorithm, it was discovered that while the robot could successfully bend around water that was offset to its left or right, it was unable to effectively evade water encountered directly in front of it. In light of this fact, the map navigation strategy was updated. Instead of performing several spirals that became increasingly smaller, the robot would perform only a single spiral along the outside of the map, ensuring that all encountered water would be offset to the robots left. To make up for the lost coverage of doing fewer spirals, a spanning algorithm was implemented. This meant after traveling a set distance the robot would stop and rotate from left to right. This is done so that when integrated with the Block Detection and Response Module, the robot can scan cubes that are not directly on its path, increasing its total map coverage. Figure 21 provides a visualization of the traversal strategy. A flowchart of Base Traversal Algorithm with Spanning is provided in Appendix Figure C.5. Further details of the motors and sensors used in this algorithm are detailed in the Appendix Table C.14.

Iterations

The Base Traversal with Spanning underwent several key iterations. These iterations are detailed in Table 16. See Appendix Table C.16 for a more detailed description of the iterations.

| Iteration Number | Description |
|---|---|
| V3.0 | The robot can return to the starting area after successfully completing one spiral, spanning not yet implemented. |
| V3.1 | The robot can return to the starting area after successfully completing one spiral. Throughout the spiral, the robot spans after travelling a set distance. |

Table 16: Iterations of Base Traversal with Spanning Algorithm

**Fast Return Home Algorithm**

If the robot encounters a cube directly in its path, a Fast Return Home algorithm is executed so that the robot can return to the starting area within the allocated trial time. During the execution of the Fast Return Home algorithm, the robot no longer scans for cubes, and no longer polls the side wall. Instead, it slightly increases the speed of the right wheel so the robot drives flush against the wall removing the necessity to scan for water. The robot would still scan the forward facing wall using the USF and when it was within the distance threshold, would turn 90° to the right. It would continue driving against the side wall and turning right when it reached the forward facing wall until it reached the starting area, at which point the trial would end. Further details of the motors and sensors used in this algorithm are detailed in Appendix Table C.17. See Appendix figure C.6 for a flowchart visualization of the algorithm.

Iterations

The Fast Return Home algorithm underwent several key iterations. These iterations are detailed in Table 17. See Appendix Table C.18 for a more detailed description of the iterations.

| Iteration Number | Description |
|---|---|
| V4.0 | The robot can successfully turn 180°, travel forward until it reaches the wall, and turn right once. |
| V4.1 | The robot can successfully turn 180°, travel forward until it reaches the wall, and turn right as many times as necessary until it returns to the starting area. The robot frequently hits water on its return. |
| V4.2 | The robot can successfully turn 180°, travel forward until it reaches the wall, and turn right as many times as necessary until it returns to the starting area. The robot rarely hits water on its return. |

Table 17: Iterations of Fast Return Home Algorithm

## Integration of Software Modules

**Integration Algorithm**

The final step to achieve a finalized software subsystem was to integrate the two modules. This was done using the Integration Algorithm. Using the Integration algorithm, the robot would begin the trial in the starting corner and traverse the map using the Base Traversal with Spanning Algorithm. If the robot detected water it would evade it using the Water Avoidance algorithm. The robot would continuously scan for cubes using the Block Detection and Approach algorithm If a cube was detected, the Classification, Capture, and Avoidance algorithm was triggered. If the cube was poop, it would be picked up and the robot would continue until it completed the spiral or encountered

another cube. If the cube was an obstacle that was not on its primary path (ie. it had travelled toward the center of the map to reach the cube), the robot would reverse and return to following the outside spiral path specified by the Base Traversal with Spanning Navigation Algorithm. If the cube was an obstacle and was on the cube's primary path, the Fast Return Home Algorithm would be executed. Once the robot returned to the starting area, the trial would end. Figure 22 provides a visualisation of the Integration algorithm in the form of a flowchart. Details of the motors and sensors used in this algorithm are detailed in Appendix Table C.19.

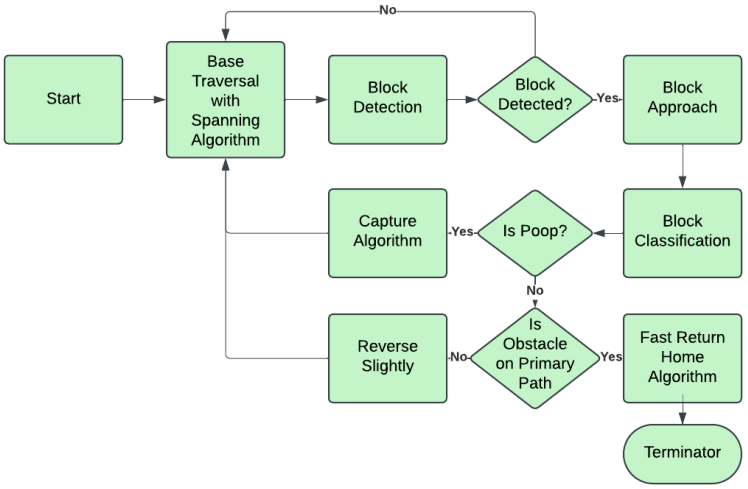



Figure 22: Flowchart Visualization of Software Module Integration

Iterations

The Integration algorithm underwent several key iterations. These iterations are detailed in Table 18. See Appendix Table C.20 for a more detailed description of the iterations.

| **Iteration Number** | **Description** |
|---|---|
| V1.0 | The robot can successfully span after travelling a set distance and pick up cubes not directly on its path. Does not yet employ the Fast Return Home algorithm |
| V1.1 | The robot can successfully span after travelling a set distance and pick up cubes not directly on its path. The robot can successfully return home using the Fast Return Home algorithm. |

Table 18: Iterations of Water Avoidance Algorithm

## E. Integration Iterative Evolution

When we integrated the hardware and the software, we only ran into very few integration problems that forced some of the changes detailed above. The first problem we ran into was in the "return home" function, our robot would get caught on the walls while returning, forcing us to build and implement a wall guard on our robot as detailed in Table 11. The second problem came from sensor error. Since the front US sensor would often miss cubes, we had to implement multiple failure recovery functions where the robot could recover itself in case of error. This involved adding special catches where the robot would move backwards whenever it was lost to allow it to recover from sensor error and find its way back as detailed in the software iterations.

# IV. Testing

Our testing strategy focused on systematically evaluating individual subsystems and their integration to ensure robust performance. Tests were planned to replicate real-world conditions, targeting critical functions like navigation, obstacle detection, water avoidance, and cube collection. We began with basic functionality tests, such as straight-line movement and sensor responsiveness, progressing to more complex scenarios like navigating around water or detecting cubes near walls.
Each test had clear objectives and consistent hardware and software configurations, enabling us to isolate and address issues effectively. Subsystem tests were followed by integration tests to assess overall system performance. This iterative approach allowed for gradual refinements, ensuring reliability and robustness in diverse scenarios.

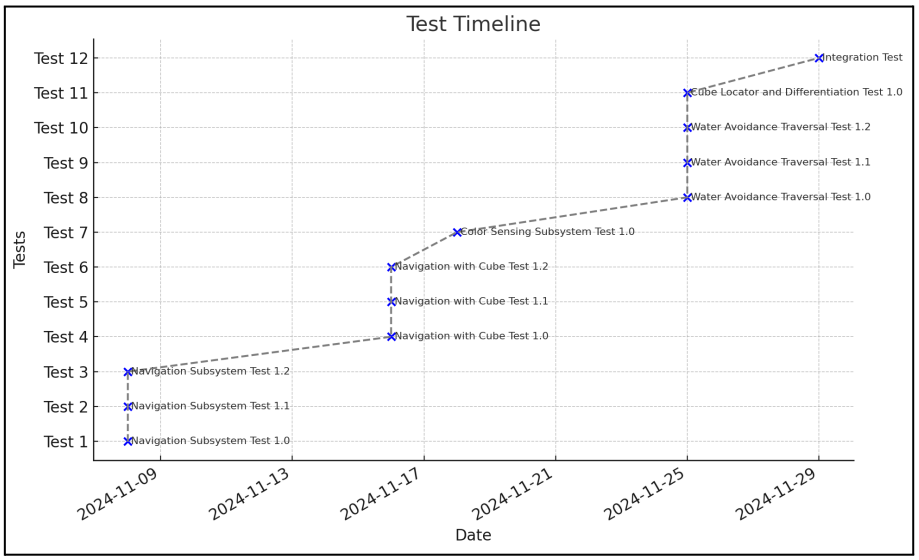


Figure 19. Test Timeline Graph

**Navigation Subsystem Test 1.0, November 8th, 2024**
**Purpose:** To validate the robot's ability to navigate a predefined path, maintaining alignment and executing 90-degree turns using ultrasonic sensors and motors.
**Procedure/Data:** see appendix Doc D.1.1: Navigation Subsystem Test 1.0
**Result:** The robot successfully followed a straight path and made a 90-degree turn but deviated from the intended route, indicating navigation software issues.
**Implications of the Test:** Software adjustments were required to enhance path adherence, Improved alignment logic and turn control became priority areas for subsequent iterations.

**Navigation Subsystem Test 1.1 is detailed in Appendix D.1.2**

**Navigation Subsystem Test 1.2, November 8th, 2024**
**Purpose:** To assess the updated software's ability to execute smooth turns and maintain accurate navigation without looping in place.
**Procedure/Data:** see appendix Doc D.1.3: Navigation Subsystem Test 1.2
**Result:** The robot accurately followed the path, realigning itself when veering off-course and executing turns as expected.
**Implications of the Test:** Confirmed that synchronization adjustments resolved prior issues, Validated the integration of side sensors for realignment, Demonstrated the system's ability to execute turns without requiring further adjustments.

**Navigation with Cube Test 1.0, November 16th, 2024**
**Purpose:** To test the robot's navigation capability while carrying cubes in the collection cage.
**Procedure/Data:** see appendix Doc D.2.1: Navigation with Cube Test 1.0

**Result:** The robot struggled to navigate smoothly, often overcorrecting and deviating from its intended path.
**Implications of the Test:** Adjustments to sensor sensitivity were necessary for smoother turns, Highlighted limitations in balancing navigation with added weight in the cube collection cage, Initiated further testing to optimize navigation algorithms for dynamic loads.

**Navigation with Cube Test 1.1 is detailed in Appendix D.2.2**

**Navigation with Cube Test 1.2, November 16th, 2024**
**Purpose:** To verify the robot's ability to maintain accurate navigation and quick corrections with cubes.
**Procedure/Data:** see appendix Doc D.2.3: Navigation with Cube Test 1.2
**Result:** The robot completed all turns successfully and adhered to the path without prolonged corrections.
**Implications of the Test:** Validated previous refinements to navigation algorithms, Demonstrated the system's stability under weight-bearing conditions, No further changes were required based on this test.

**Color Sensing Subsystem Test 1.0, November 18th, 2024**
**Purpose:** To evaluate the color sensors' ability to detect and classify surfaces and objects accurately.
**Procedure/Data:** see appendix Doc D.3.1: Color Sensing Test 1.0
**Result:** The robot successfully identified most colors but occasionally misclassified overlapping colors. Additionally, the color sensor failed to detect cubes when positioned at a 45-degree angle, causing misclassification or failure to identify the object entirely.
**Implications of the Test:** Sensor accuracy for edge cases (e.g., color overlap and 45-degree angle detection) was highlighted as a limitation, Refinements in color classification algorithms were prioritized to address angle-specific failures, Expanded the testing scope to include diverse environmental conditions and sensor placements.

**Cube Locator and Differentiation Test 1.0, November 25th, 2024**
**Purpose:** To assess the robot's ability to detect cubes and differentiate between obstacles and "poop."
**Procedure & Data:** See Appendix Doc D.5.1: Cube Locator and Differentiation Test 1.0

**Result:** The robot performed well in open areas but struggled near walls and when the cube is at an angle of 45 degree and ran into both obstacles and poop.
**Implications of the Test:** Sensor accuracy for edge cases (e.g., color overlap and 45-degree angle detection) was highlighted as a limitation, Adjusted ultrasonic sensor detection range for better performance near walls, Improved the color sensor's classification algorithm for edge cases, Tested additional placements in tighter spaces to ensure robust detection.

**Water Avoidance Traversal with Rotation Test 1.0, November 25th, 2024**
**Purpose:** To ensure the robot avoids water, rotates properly, and completes its path.
**Procedure/Data:** see appendix Doc D.4.1: Water Avoidance Traversal with Rotation Test 1.0
**Result:** The robot avoided water and navigated back to the starting position successfully.
**Implications of the Test:** Optimized rotation logic to ensure smoother transitions after avoiding obstacles, Identified the need for robust water detection under varying environmental conditions, Established a baseline for further enhancements in reaction speed and path correction.

**Water Avoidance Traversal with Rotation Test 1.1, November 25th, 2024**
**Purpose:** To test the robot's behavior when encountering a cube close to the wall and to evaluate its ability to navigate without colliding into the wall.
**Procedure & Data:** See appendix Doc D.4.2: Water Avoidance Traversal with Rotation Test 1.1
**Result:** The robot detected cubes correctly but collided with the wall when the cube was near the wall.

**Implications of the Test:** Enhanced algorithms for distinguishing between cubes and walls were required, Increased sensitivity of side sensors to improve proximity detection, Introduced logic for path adjustments post-cube detection.

**Water Avoidance Traversal with Rotation Test 1.2, November 25th, 2024**
**Purpose:** To evaluate navigation performance when encountering water directly in its path.
**Procedure/Data:** see appendix Doc D.4.3: Water Avoidance Traversal with Rotation Test 1.2
**Result:** The robot detected water but failed to adjust its path, leading to a collision.
**Implications of the Test:** Prioritized enhancements to water detection algorithms for immediate path adjustments, Implemented an emergency stop mechanism for head-on water detection failures, Refined movement speed during inner loop navigation for more precise reactions.

**Integration Test 1.0, November 28th, 2024**
**Purpose:** To ensure all subsystems (navigation, water avoidance, cube detection, and collection) work cohesively - Link to practise test.
**Procedure/Data:** see appendix Doc D.6.1: Integration Test 1.0
**Result:** The robot demonstrated strong performance across most tasks but struggled with detecting cubes near walls, detecting cubes at a 45-degree angle, and returning to the trash area.
**Implications of the Test:** Refined navigation logic to reattempt grid coverage after encountering obstacles, Adjusted detection range for cubes near walls to reduce errors, Improved the return-to-trash algorithm and addressed limitations in color sensor angle detection.

**Integration Test 1.1, December 3rd, 2024**
**Purpose:** To ensure all subsystems (navigation, water avoidance, cube detection, and collection) work cohesively.
**Procedure/Data:** see appendix Doc D.6.2: Integration Test 1.1
**Result:** The robot demonstrated partial success in detecting and scanning cubes, with 50% accuracy for both tasks. While 66% of the cubes were successfully picked up, the robot reliably returned to the trash area. Performance gaps remain in handling detection and scanning consistency.
**Implications of the Test:** Detection and scanning algorithms require further refinement to improve accuracy across all cube types. Adjustments to pickup logic can ensure higher collection success rates. The return-to-trash functionality remains robust, but enhanced navigation and obstacle handling may be necessary to improve overall task cohesion.

# V. Performance

## A. System Capabilities and Performance

Based on the problem specifications and constraints our robot was able to meet most of the capabilities of the problem set. It could successfully traverse the outside section of the grid, covering approximately 65% of the available space within the bounds of the red gridlines when taking into account the space taken up by water. Using its front US sensor it could successfully detect the cubes that were in front of it and then use the right color sensor to analyze its color. It would find cubes and adjust itself to place the cube under the color sensor successfully for approximately 77% of the cubes in its path. Once the cube was under the sensor, it would identify the correct color with a 100% success rate. Once identified, the robot can successfully pick up poop and avoid touching the obstacle with a perfect success rate based on trial tests. The robot never once fell into water in our final tests and never got lost, returning home with a 100% success rate. Although it never happened in the tests since 6 cubes were never in its path, the robot can successfully hold 6 cubes in its cage simultaneously. Note that apart from the demo, the following test runs weren't constrained in time and represent the capabilities of the robot when given sufficient time to complete all tasks. The Demo and one of the test runs can be viewed through these links: Demo - Test (see run #5 below).

| Run # | Poops Picked Up | Poops Dropped Off | Cubes Detected in Path | Returned Home | Obstacle Collisions | Other penalties (Water/Touch) | Total Points |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3/3 | yes | 1 | 0 | 9.5 |
| 2 | 3 | 0 | 5/6 | yes | 1 | 0 | 8.5 |
| 3 | 1 | 1 | 2/3 | yes | 0 | 0 | 8 |
| 4 | 4 | 4 | 5/7 | yes | 1 | 0 | 13.5 |
| 5 | 4 | 4 | 5/8 | yes | 2 | 0 | 13 |
| 6 | 3 | 3 | 5/6 | yes | 0 | 0 | 12 |
| Demo | 3 | 0 | 3/4 | yes | 1 | 0 | 8.5 |

Table 22: Robot Performance in 7 tests done on December 3rd and the December 4th Demo

## B. System Usage

To use the system, you have to first place the robot in the starting zone with the right US sensor facing the closest wall, the robot facing up (based on the original map image). The back of the robot should touch the wall and the left wheel should be in line with the edge of the zone (see Appendix E.1). The front US sensor should be set in its down position (see Appendix E.2). You have to connect to the robot using the BrickPi's Hotspot and run the program called final_integration.py. You might need to wait a little more than 3 minutes for the robot to complete its entire sweep and successfully deposit the poops it has collected.

## C. Performance Limitations

The primary performance limitation of our prototype is the speed at which it traverses the map. As the robot navigates its environment, it regularly spans inwards and approaches every block it detects. This means that the more blocks the robot encounters, the longer it will take to complete a single spiral. During the demo, the robot was unable to complete its full spiral due to this limitation meaning it was unable to deliver the 3 poop cubes it had collected during the trial to the trash area.

Another limitation of the prototype was its inability to detect cubes in the center of the map. The base traversal with spanning algorithm allowed the robot to scan up to 66% of the map surface by traversing the outside of the map while occasionally scanning inwards. This meant the robot was not able to scan cubes located in the remaining 33% of the map, limiting the maximum number of cubes the robot would be able to collect during its trial.

The final major limitation of the robot had to do with its Cube Scanning Algorithm. Due to the unreliability of the front ultrasonic sensor the robot occasionally did not detect cubes that were directly in front of it. This caused the robot to often hit obstacles or fail to collect cubes that were directly on its path. While this problem was mitigated throughout the system level tests it was not fully eliminated and caused the robot to hit one obstacle during the demo.

# VI. Appendix

## A. Introduction

Doc A.1. Team Contract & Shared Goals Document

Team Contract

**Commitment to Weekly Hours**: Each team member agrees to dedicate 9 hours per week to the project, as specified by the design week guidelines. Any adjustments to weekly contributions should be communicated and documented within the team.

**Regular Communication**: Team members will maintain open lines of communication through MS Teams and attend all scheduled team meetings. Members will notify the team of any conflicts ahead of time and provide updates on their progress.

**Documentation and File Organization**: Each team member agrees to contribute to maintaining organised and up-to-date documentation on MS Teams, including timesheets, meeting agendas, and design documents. All final weekly deliverables will be uploaded on time for review.

**Respect for Individual Roles and Contributions**: All team members will respect each other's roles as outlined in the team structure. Members are encouraged to support others outside their role when appropriate but will prioritise their assigned tasks and responsibilities.

**Commitment to Continuous Improvement**: Each member will actively engage in design reviews and feedback sessions to enhance both the design and individual performance. The team will reflect on feedback from TAs and peers to improve the design iteratively.

Shared Goals

**Strengthen Technical Proficiency**: Gain hands-on experience with the LEGO + BrickPi kits and become proficient in programming and controlling robotic systems, particularly for environmental navigation and object collection tasks.

**Maximise Collaboration and Role Clarity**: Develop a strong understanding of each team member's unique skills and assigned roles to foster seamless collaboration. Ensure each member's strengths are utilised effectively in both leadership and technical contributions.

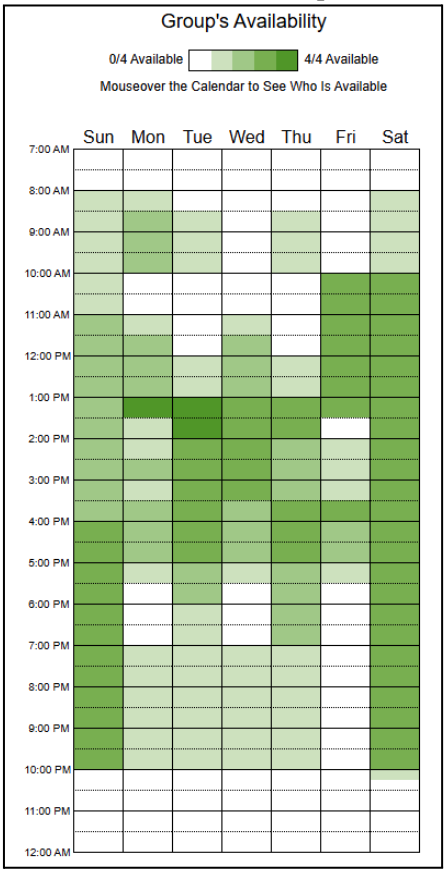
**Embrace the Engineering Design Process**: Build a comprehensive understanding of the engineering design process from conceptualization to testing. Emphasise iterative improvement, problem-solving, and documentation to deliver a clear, reproducible solution.

**Deliver a Robust, Client-Focused Solution**: Design a prototype that meets the client's requirements for a robotic dog poop cleaner, ensuring the robot's efficiency in object collection, obstacle avoidance, and trash disposal within the specified constraints.

**Promote a Culture of Accountability and Support**: Establish regular check-ins to monitor progress, share challenges, and support each other in overcoming technical or scheduling obstacles. Use these sessions to uphold accountability and collectively address any project roadblocks.

Signatures:

Table A.1 - When2meet Group Availability



## Untitled Gantt Project

Nov. 4, 2024

http://

| | |
|---|---|
| **Project manager** | |
| **Project dates** | Oct. 28, 2024 - Dec. 5, 2024 |
| **Completion** | 0% |
| **Tasks** | 69 |
| **Resources** | 0 |

Figure A.1.a Initial Project Gantt Chart

## Untitled Gantt Project

Nov. 4, 2024

2

## Tasks

| Name | Begin date | End date |
|---|---|---|
| Week 1 | 2024-10-28 | 2024-11-03 |
| Update Timesheet | 2024-10-28 | 2024-11-03 |
| Budget Documentation | 2024-10-28 | 2024-10-31 |
| Budgeting | 2024-10-28 | 2024-10-28 |
| Gantt Chart Creation | 2024-10-28 | 2024-10-28 |
| Week 1 Meeting Agenda | 2024-10-28 | 2024-10-28 |
| Initial meeting | 2024-10-29 | 2024-10-29 |
| Brainstorming Design Ideas | 2024-10-29 | 2024-10-29 |
| Team Contract | 2024-10-29 | 2024-10-29 |
| Brainstorming Documentation | 2024-10-30 | 2024-10-30 |
| Week 1 Meeting Minutes | 2024-10-29 | 2024-10-29 |
| Requirement Writing | 2024-10-30 | 2024-10-30 |
| Requirement Review | 2024-10-31 | 2024-10-31 |
| Outline Project Contraints | 2024-10-31 | 2024-10-31 |
| Gantt Chart Review | 2024-10-31 | 2024-10-31 |
| Model Mobile Base with Color Sensors | 2024-11-01 | 2024-11-01 |
| Basic Spiral Traversal Code do Robot (Ignore obstacles) | 2024-11-02 | 2024-11-02 |
| Test Traversal Code (Ensure entire surface is covered) | 2024-11-03 | 2024-11-03 |
| Test Ultrasonic Sensors for Positioning Accuracy | 2024-11-03 | 2024-11-03 |
| Week 2 | 2024-11-04 | 2024-11-11 |
| Update Timesheets | 2024-11-04 | 2024-11-11 |
| Weekly Progress Meeting | 2024-11-04 | 2024-11-04 |
| Initial Design Review / Evaluation | 2024-11-04 | 2024-11-04 |
| Model Cube "Pickup" Mechanism | 2024-11-05 | 2024-11-05 |
| Model raising and lowering of the ultrasonic sensor | 2024-11-05 | 2024-11-05 |
| Implement mobile base | 2024-11-06 | 2024-11-06 |
| Implement Cube "Pickup" Mechanism | 2024-11-06 | 2024-11-06 |

Figure A.1.b Initial Project Gantt Chart

## Tasks

| Name | Begin date | End date |
|---|---|---|
| Write Code for Lowering and Raising of US Sensor | 2024-11-07 | 2024-11-07 |
| Implement raising and lowering of the ultrasonic sensor | 2024-11-07 | 2024-11-07 |
| Add Water Consideration to Traversal Code | 2024-11-07 | 2024-11-07 |
| Test Water Avoidance (Cover all green while avoiding water) | 2024-11-08 | 2024-11-08 |
| Week 2 Meeting Agenda | 2024-11-08 | 2024-11-08 |
| Week 2 Meeting Minutes | 2024-11-08 | 2024-11-08 |
| Finalize Budget | 2024-11-09 | 2024-11-09 |
| Hardware Design Document | 2024-11-09 | 2024-11-09 |
| Software Design Document | 2024-11-09 | 2024-11-09 |
| Model/System Description | 2024-11-10 | 2024-11-10 |
| Test Color Sensors to Measure Floor Colour at Cube Height | 2024-11-10 | 2024-11-10 |
| Test Color sensors for cubes | 2024-11-10 | 2024-11-10 |
| Initial Integration Test for Sensors and Movement | 2024-11-11 | 2024-11-11 |
| Week 2 Full Testing Documentation | 2024-11-11 | 2024-11-11 |
| Report Test Results to Software Team | 2024-11-11 | 2024-11-11 |
| Week 3 | 2024-11-12 | 2024-11-18 |
| Weekly Progress Meeting | 2024-11-12 | 2024-11-12 |
| Discuss Current Problems in Design and Potential Solutions | 2024-11-12 | 2024-11-12 |
| Implement Full Navigation Algorithm For Robot | 2024-11-13 | 2024-11-13 |
| Test Navigation System (Percentage of poop collected) | 2024-11-14 | 2024-11-14 |
| Week 3 Meeting Agenda | 2024-11-14 | 2024-11-14 |
| Week 3 Meeting Minutes | 2024-11-14 | 2024-11-14 |
| Full Initial Hardware Integration Test | 2024-11-15 | 2024-11-16 |
| Full Initial Software Integration Test | 2024-11-15 | 2024-11-16 |
| Create Project Report Template | 2024-11-15 | 2024-11-17 |
| Reflect on Integration Tests / Plan Upcoming Tests | 2024-11-17 | 2024-11-17 |
| Bug Fixes on Overall Code Design (Unsure where yet) | 2024-11-17 | 2024-11-17 |

Figure A.1.c Initial Project Gantt Chart



## Tasks

| Name | Begin date | End date |
|---|---|---|
| Arrange cable management | 2024-11-18 | 2024-11-18 |
| Week 4 | 2024-11-19 | 2024-11-26 |
| Final Integration Testing (Full system SW and HW) | 2024-11-19 | 2024-11-19 |
| Centralize Everyone's Contributions into the Report | 2024-11-20 | 2024-11-20 |
| Final HW Design Documentation | 2024-11-21 | 2024-11-22 |
| Final SW Design Documentation | 2024-11-21 | 2024-11-22 |
| Final Testing Documentation | 2024-11-22 | 2024-11-23 |
| Final Bug Fixes | 2024-11-23 | 2024-11-25 |
| Prepare for Demo | 2024-11-23 | 2024-11-25 |
| Report Buffer | 2024-11-25 | 2024-11-26 |
| Week 5 | 2024-11-27 | 2024-12-04 |
| Final test runs for edge cases (low battery, etc.) | 2024-11-27 | 2024-12-01 |
| Final Report review | 2024-11-28 | 2024-12-01 |
| Work Buffer | 2024-12-02 | 2024-12-04 |
| Demo | 2024-12-05 | 2024-12-05 |

Figure A.1.d Initial Project Gantt Chart

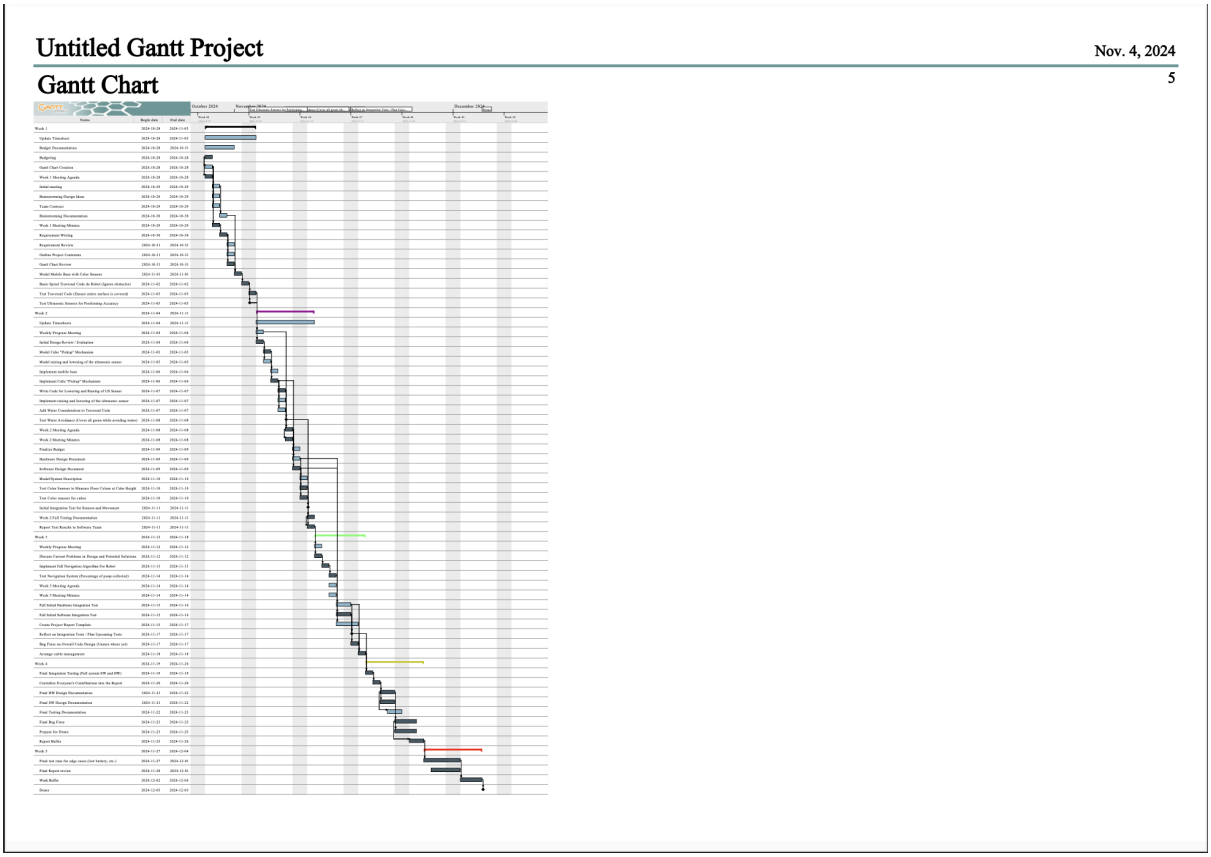



Figure A.1.e

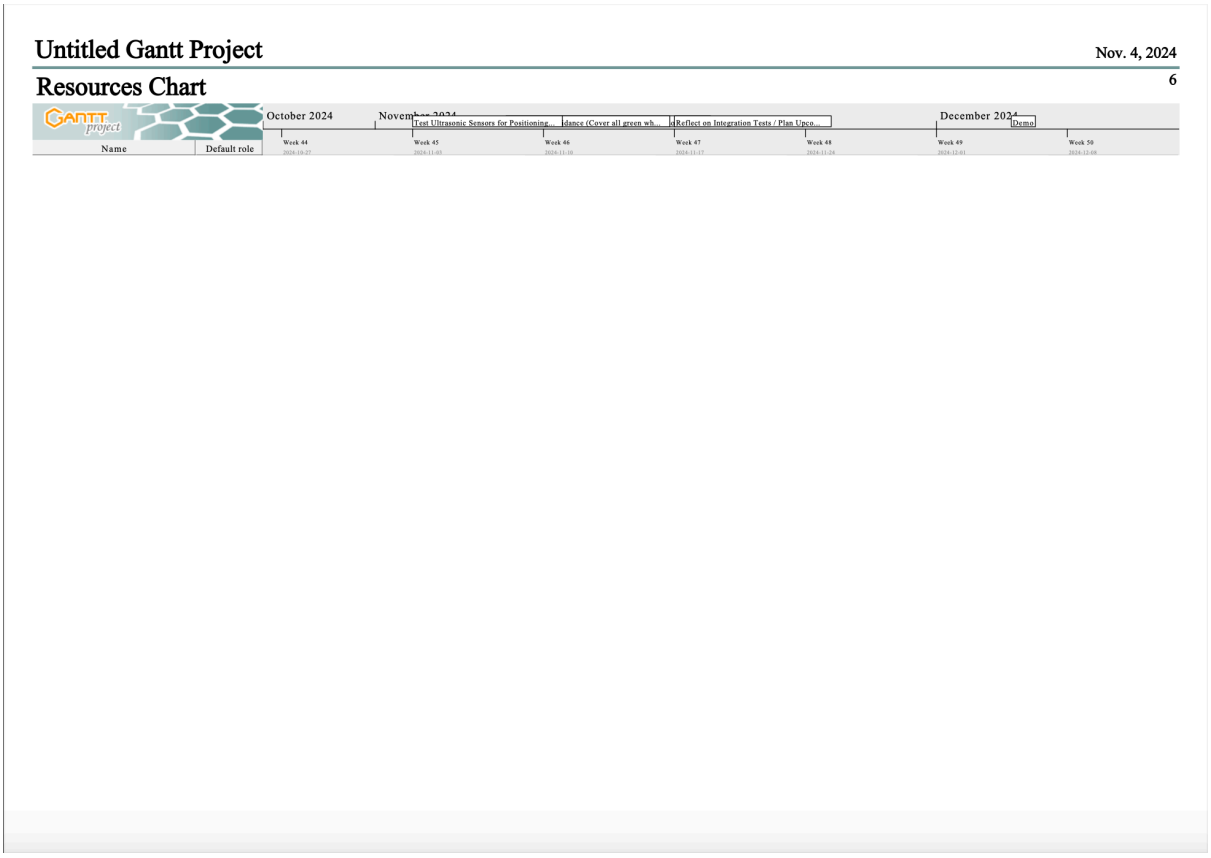



Figure A.1.f

| | Hours Per Person: Marrec | Will | Garrett | Nic | Karim | Katrina | Total Hours |
|---|---|---|---|---|---|---|---|
| Total For Each Person: | 45 | 45 | 45 | 45 | 45 | 44.5 | 269.5 |
| **Milestone/Week 1: Intial Plan Created** | | | | | | | |
| Initial Meeting | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 9 |
| Brainstorming Design Ideas | 2.5 | 2.5 | 2.5 | 2.5 | | | 10 |
| Budgeting | 1.5 | | | 1.5 | | | 3 |
| Brainstorming Documentation | | | | 2 | | | 2 |
| Gantt Chart Creation | | | | | 4 | | 4 |
| Team Contract | | | | | | 1 | 1 |
| Model Mobile Base With Color Sensors | | | | 2 | 1 | | 3 |
| Week 1 Meeting Agenda | | 1 | | | | | 1 |
| Week 1 Meeting Minutes | | 0.5 | | | | | 0.5 |
| Requirement Writing | | 2.5 | | | | | 2.5 |
| Requirement Review | | | | | | 1.5 | 1.5 |
| Outline Project Constraints | | | | | | | 0 |
| Budget Documentation | 1 | | | | | | 1 |
| Gantt Chart Review | | | 2 | | | | 2 |
| Basic Spiral Traversal Code for Robot (Ignore obstacles) | 2 | | | | | | 2 |
| Test Traversal Code (Ensure entire surface is covered) | | | | | | 1.5 | 1.5 |
| Test Ultrasonic Sensors for Positioning Accuracy | | | | | | 1 | 1 |
| Update Timesheet | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 1.5 |
| | | | | | | | 0 |
| | | | | | | | 0 |
| **Milestone/Week 2: First Semi-Functional Product is Built** | | | | | | | 0 |
| Weekly Progress Meeting | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| Initial Design Review / Evaluation | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 3 |
| Model Cube "Pickup" Mechanism | | | | 1 | 1 | | 2 |
| Model raising and lowering of the ultrasonic sensor | | | | 1 | | | 1 |
| Implement mobile base | | | | 2 | 0.5 | | 2.5 |
| Implement Cube "Pickup" Mechanism | | | | 1 | 1 | | 2 |
| Implement raising and lowering of the ultrasonic sensor hardware | | | | 1 | 1 | | 2 |
| Add Water Consideration to Traversal Code | 2.5 | 1 | | | | | 3.5 |
| Test Water Avoidance (Cover all green while avoiding water) | | | | | 2 | 2 | 4 |
| Week 2 Meeting Agenda | | 1 | | | | | 1 |
| Week 2 Meeting Minutes | | 0.5 | | | | | 0.5 |
| Update Timesheets | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 1.5 |
| Finalize Budget | | | | | 1.5 | | 1.5 |
| Hardware Design Document | | | | 3 | | | 3 |
| Software Design Document | 1.5 | 1 | 1.5 | | | | 4 |

Figure A.2.a Initial Project budget

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Model/System Description | 1 | 2 | | | | | 3 |
| Week 2 Full Testing Documentation | | | | | 3 | 3 | 6 |
| Report Test Results to Software Team | 1 | 1 | 1 | | | 1 | 4 |
| | | | | | | | 0 |
| | | | | | | | 0 |
| **Milestone/Week 3: Prototype is Finalized and Optimization has begun** | | | | | | | 0 |
| Test Color sensors for cubes | | | | | | 2 | 2 |
| Test Color Sensors to Measure Floor Colour at Cube Height | | | | | | 2 | 2 |
| Write Code for Lowering and Raising of US Sensor | 1.5 | | 1.5 | | | | 3 |
| Initial Integration Test for Sensors and Movement | | | | | 1.5 | 1.5 | 3 |
| Weekly Progress Meeting | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| Design Review Presentation Create | | | 3 | | 0.5 | | 3.5 |
| Discuss Current Problems in Design and Potential Solutions | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 3 |
| Implement Full Navigation Algorithm For Robot | 4 | 4 | | | | | 8 |
| Test Navigation System (Percentage of poop collected) | | | | | 1.5 | 2 | 3.5 |
| Full Initial Hardware Integration Test | | | | 2 | 1 | | 3 |
| Full Initial Software Integration Test | 1 | | | | 1 | 1 | 3 |
| Reflect on Integration Tests / Plan Upcoming Tests | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 3 |
| Bug Fixes on Overall Code Design (Unsure where yet) | 2.5 | 2.5 | 2.5 | | | | 7.5 |
| Week 3 Meeting Agenda | | 1 | | | | | 1 |
| Week 3 Meeting Minutes | | 0.5 | | | | | 0.5 |
| Arrange cable management | | | | 1 | | | 1 |
| Create Project Report Template | | | 1.5 | | | | 1.5 |
| | | | | | | | 0 |
| **Milestone/Week 4: Report Draft is Complete and Robot is Demo Ready** | | | | | | | 0 |
| Final Integration Testing (Full system SW and HW) | | | | | 2 | 2 | 4 |
| Centralize Everyone's Contributions into the Report | | | 3 | | | | 3 |
| Team Design Reviews (Nov 20) | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 9 |
| Final HW Design Documentation | | | | 3 | 2 | | 5 |
| Final SW Design Documentation | | | 3 | | | | 3 |
| Final Testing Documentation | | | | | | 3 | 3 |
| Final Bug Fixes | 3 | 3 | 3 | | | | 9 |
| Prepare for Demo | 2 | 2 | 2 | 2 | 2 | 2 | 12 |
| Report Buffer | 2 | 3 | 2 | 3 | 3 | 2 | 15 |
| | | | | | | | 0 |
| | | | | | | | |
| **Milestone/Week 5: Report is Finalized and Demo is performed** | | | | | | | 0 |
| Final Report review | 2 | 2 | 2.5 | 2 | 2 | 2 | 12.5 |
| Work Buffer | 4 | 4 | 4 | 5 | 3.5 | 4 | 24.5 |
| Final test runs for edge cases (low battery, etc.) | 2 | 2 | 2 | 2 | 2 | 2 | 12 |

Figure A.2.b Initial Project Budget

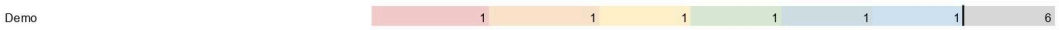



Figure A.2.c

## ECSE 211 Final Project

Dec. 4, 2024

## Marwan Fan Club

http://

| | |
|---|---|
| **Project manager** | |
| **Project dates** | Oct. 28, 2024 - Dec. 4, 2024 |
| **Completion** | 0% |
| **Tasks** | 99 |
| **Resources** | 6 |

Figure A.3.a

## Tasks

| Name | Begin date | End date | Resources |
|---|---|---|---|
| Week 1: Inital Plan Created | 2024-10-28 | 2024-11-03 | |
| Update Timesheet | 2024-10-28 | 2024-11-03 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Budget Documentation | 2024-10-28 | 2024-10-31 | Marrec |
| Create Project Budget | 2024-10-28 | 2024-10-28 | Marrec, Nic |
| Budgeting | 2024-10-28 | 2024-10-28 | Marrec, Nic |
| Gantt Chart Creation | 2024-10-28 | 2024-10-28 | Karim |
| Week 1 Meeting Agenda | 2024-10-28 | 2024-10-28 | William |
| Initial meeting | 2024-10-29 | 2024-10-29 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Brainstorming Design Ideas | 2024-10-29 | 2024-10-29 | Marrec, William, Garrett, Nic |
| Team Contract | 2024-10-29 | 2024-10-29 | Katrina |
| Brainstorming Documentation | 2024-10-30 | 2024-10-30 | Nic |
| Week 1 Meeting Minutes | 2024-10-29 | 2024-10-29 | William |
| Requirement Writing | 2024-10-30 | 2024-10-30 | William |
| Requirement Review | 2024-10-31 | 2024-10-31 | Katrina |
| Outline Project Contraints | 2024-10-31 | 2024-10-31 | Katrina |
| Gantt Chart Review | 2024-10-31 | 2024-10-31 | Garrett |
| Model Mobile Base with Color Sensors | 2024-11-01 | 2024-11-01 | Nic, Karim |
| Basic Spiral Traversal Code do Robot (Ignore obstacles) | 2024-11-02 | 2024-11-02 | Marrec |
| Test Traversal Code (Ensure entire surface is covered) | 2024-11-03 | 2024-11-03 | Katrina |
| Test Ultrasonic Sensors for Positioning Accuracy | 2024-11-03 | 2024-11-03 | Katrina |
| Week 2: First Semi-Functional Product is Built | 2024-11-04 | 2024-11-11 | |
| Update Timesheets | 2024-11-04 | 2024-11-11 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Weekly Progress Meeting | 2024-11-04 | 2024-11-04 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Initial Design Review / Evaluation | 2024-11-04 | 2024-11-04 | Marrec, William, Garrett, Nic, Karim, Katrina |

Figure A.3.b

## Tasks

| Name | Begin date | End date | Resources |
|---|---|---|---|
| Model Cube "Pickup" Mechanism | 2024-11-05 | 2024-11-05 | Nic, Karim |
| Update hardware sensor placement | 2024-11-05 | 2024-11-05 | Nic |
| Increase hardware carrying capacity | 2024-11-06 | 2024-11-06 | Nic |
| Model raising and lowering of the ultrasonic sensor | 2024-11-05 | 2024-11-05 | Nic |
| Implement mobile base | 2024-11-06 | 2024-11-06 | Nic, Karim |
| Implement Color Classification Algorithm | 2024-11-05 | 2024-11-05 | William |
| Write Code for Lowering and Raising of US Sensor | 2024-11-07 | 2024-11-07 | Marrec, Garrett |
| Implement raising and lowering of the ultrasonic sensor | 2024-11-07 | 2024-11-07 | Nic, Karim |
| Add Water Detection to Traversal Code | 2024-11-06 | 2024-11-06 | Marrec, William |
| Add Water Avoidance to Traversal Code | 2024-11-07 | 2024-11-07 | Marrec, William, Garrett |
| Test Water Avoidance (Cover all green while avoiding water) | 2024-11-08 | 2024-11-08 | Karim, Katrina |
| Week 2 Meeting Agenda | 2024-11-08 | 2024-11-08 | William |
| Week 2 Meeting Minutes | 2024-11-08 | 2024-11-08 | William |
| Update Budget | 2024-11-09 | 2024-11-09 | Karim |
| Hardware Design Document | 2024-11-09 | 2024-11-09 | Nic |
| Software Design Document | 2024-11-09 | 2024-11-09 | Marrec, William, Garrett |
| Model/System Description | 2024-11-10 | 2024-11-10 | Marrec, William |
| Test Color Sensors to Measure Floor Colour at Cube Height | 2024-11-10 | 2024-11-10 | Katrina |
| Test Color sensors for cubes | 2024-11-10 | 2024-11-10 | Karim, Katrina |
| Initial Integration Test for Sensors and Movement | 2024-11-11 | 2024-11-11 | Karim, Katrina |
| Week 2 Full Testing Documentation | 2024-11-11 | 2024-11-11 | Karim, Katrina |
| Report Test Results to Software Team | 2024-11-11 | 2024-11-11 | Marrec, William, Garrett |
| Week 3: Prototype is Finalized | 2024-11-12 | 2024-11-18 | |
| Test Color sensors for cubes | 2024-11-12 | 2024-11-12 | Katrina |

Figure A.3.c

## Tasks

| Name | Begin date | End date | Resources |
|---|---|---|---|
| Test Color Sensors to Measure Floor Colour at Cube Height | 2024-11-12 | 2024-11-12 | Katrina |
| Create Design Review Presentation Draft | 2024-11-12 | 2024-11-12 | Nic, Karim |
| Weekly Progress Meeting | 2024-11-12 | 2024-11-12 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Initial Integration Test for Sensors and Movement | 2024-11-13 | 2024-11-13 | Karim, Katrina |
| Update Design Review Presentation based on feedback | 2024-11-13 | 2024-11-13 | Nic |
| Discuss Current Problems in Design and Potential Solutions | 2024-11-12 | 2024-11-12 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Implement Full Navigation Algorithm For Robot | 2024-11-13 | 2024-11-13 | Marrec, William |
| Test Navigation System (Percentage of poop collected) | 2024-11-14 | 2024-11-14 | Karim, Katrina |
| Week 3 Meeting Agenda | 2024-11-14 | 2024-11-14 | William |
| Week 3 Meeting Minutes | 2024-11-14 | 2024-11-14 | William |
| Full Initial Hardware Integration Test | 2024-11-15 | 2024-11-16 | Nic, Karim |
| Full Initial Software Integration Test | 2024-11-15 | 2024-11-16 | Karim, Katrina |
| Create Project Report Template | 2024-11-15 | 2024-11-17 | Garrett |
| Reflect on Integration Tests / Plan Upcoming Tests | 2024-11-17 | 2024-11-17 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Bug Fixes on Overall Code Design | 2024-11-17 | 2024-11-17 | Marrec, William, Garrett |
| Implement Cube "Pickup" Mechanism | 2024-11-17 | 2024-11-17 | Nic, Karim |
| Create Project Report Template | 2024-11-17 | 2024-11-17 | Nic |
| Arrange cable management | 2024-11-18 | 2024-11-18 | Nic |
| Implement Cube Detection and Approach Algorithm | 2024-11-18 | 2024-11-18 | Marrec |
| Week 4: Report Draft and Integration Testing is Complete | 2024-11-19 | 2024-11-26 | |
| Final Integration Testing (Full system SW and HW) | 2024-11-19 | 2024-11-19 | Karim, Katrina |
| Centralize Everyone's Contributions into the Report | 2024-11-20 | 2024-11-20 | Garrett |
| Team Design Reviews | 2024-11-20 | 2024-11-20 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Bug Fixes In Cube Pickup | 2024-11-20 | 2024-11-20 | Marrec, Garrett |

Figure A.3.d

## Tasks

| Name | Begin date | End date | Resources |
|---|---|---|---|
| Cube Avoidance algorithm | 2024-11-21 | 2024-11-21 | Marrec |
| Write Return to Trash Algorithm | 2024-11-21 | 2024-11-21 | William |
| Update water avoidance to avoid all water | 2024-11-22 | 2024-11-22 | William, Nic |
| Revamp Navigation Algorithm | 2024-11-22 | 2024-11-22 | Marrec, Garrett |
| Optimize Color Classification Algorithm | 2024-11-22 | 2024-11-22 | William |
| Test Revamped Navigation Algorithm | 2024-11-23 | 2024-11-23 | Karim, Katrina |
| Final Hardware Optimizations | 2024-11-24 | 2024-11-24 | Nic |
| Work Buffer | 2024-11-25 | 2024-11-26 | |
| Week 5 | 2024-11-27 | 2024-12-03 | |
| Final Complete Integration Test | 2024-11-27 | 2024-12-01 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Finalize Budget | 2024-11-27 | 2024-11-27 | Garrett |
| Final HW Design Documentation | 2024-11-27 | 2024-11-27 | Nic, Karim |
| Final SW Design Documentation | 2024-11-27 | 2024-11-27 | Garrett |
| Final Bug Fixes | 2024-11-27 | 2024-11-27 | Marrec, William, Garrett |
| Final Testing Documentation | 2024-11-28 | 2024-11-28 | Katrina |
| Team Strategy Meeting | 2024-11-28 | 2024-11-28 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Finalize Gantt Chart | 2024-11-28 | 2024-11-28 | Karim |
| Organize and Compile Test Data into report | 2024-11-29 | 2024-11-29 | Karim, Katrina |
| Project Management Section of Report | 2024-11-29 | 2024-11-29 | Marrec |
| Add Software Design and Iterations Into report | 2024-11-30 | 2024-11-30 | William |
| Add Hardware Designs and Iterations into report | 2024-11-30 | 2024-11-30 | Nic |
| Create Professional Sketches of Robot for Report | 2024-11-30 | 2024-11-30 | Karim |
| Final test runs for edge cases (low battery, etc.) | 2024-11-30 | 2024-11-30 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Prepare for Demo | 2024-11-29 | 2024-12-01 | Marrec, William, Garrett, Nic, Karim, Katrina |

Figure A.3.e

## Tasks

| Name | Begin date | End date | Resources |
|---|---|---|---|
| Final Report review | 2024-12-01 | 2024-12-01 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Work Buffer | 2024-12-02 | 2024-12-03 | Marrec, William, Garrett, Nic, Karim, Katrina |
| Demo | 2024-12-04 | 2024-12-04 | Marrec, William, Garrett, Nic, Karim, Katrina |

Figure A.3.f

## Resources

| Name | Default role |
|---|---|
| Marrec | Software Team Leader |
| William | Design Documentation Manager and Software Engineer |
| Garrett | Design Team Manager and Software Engineer and Test Engineer |
| Nic | Hardware Team Leader and Test Engineer |
| Karim | Lead Draughtsman, Hardware Engineer and Test Engineer |
| Katrina | Testing Team Leader |

Figure A.3.g

## Resources Chart

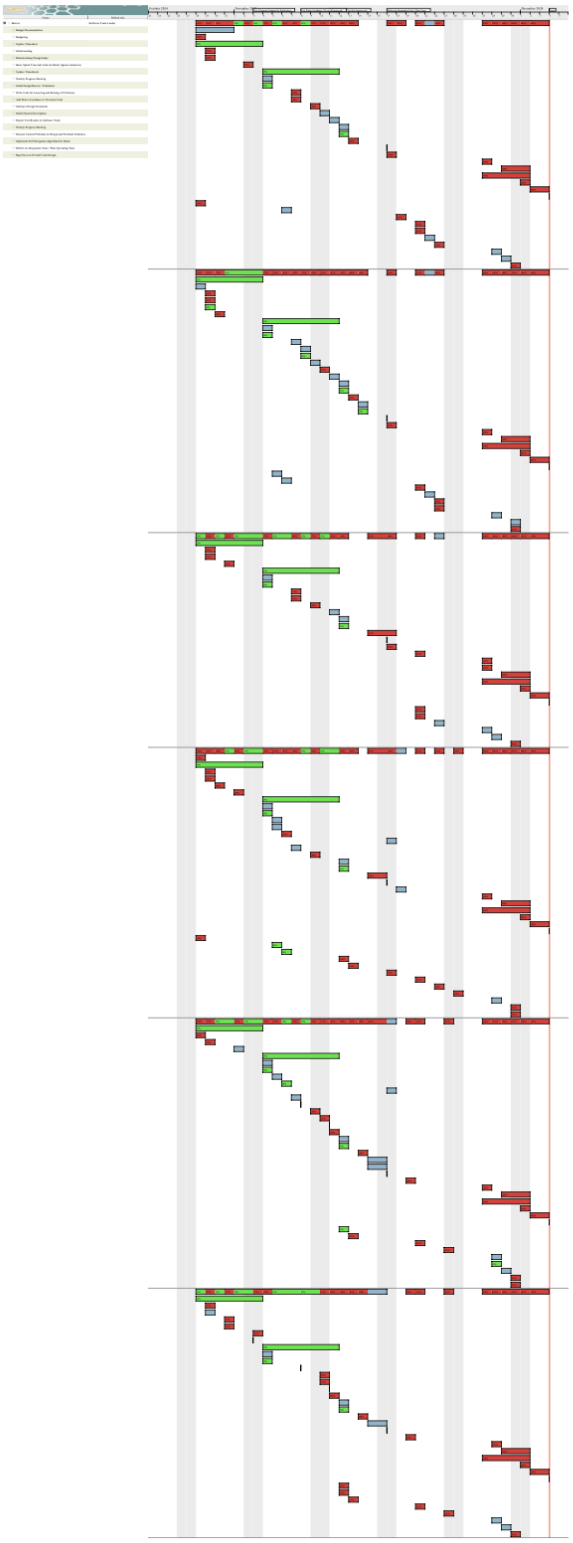
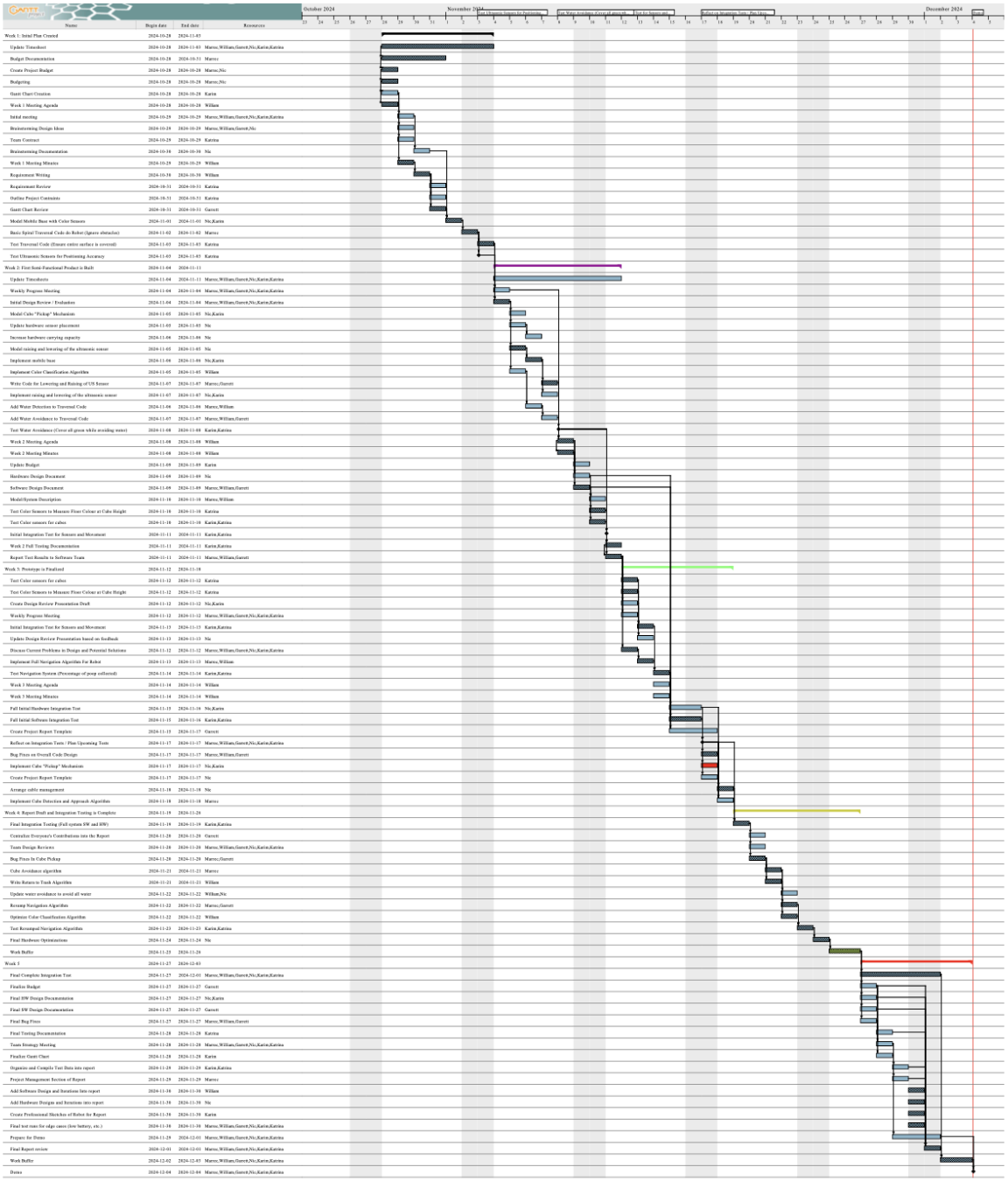


Figure A.3.h

## Gantt Chart





| Name | Begin date | End date | Resources |
|---|---|---|---|
| Week 1: Initial Plan Created | 2024-10-28 | 2024-11-03 | |
| Update Timesheet | 2024-10-28 | 2024-11-03 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Budget Documentation | 2024-10-28 | 2024-10-31 | Marrec |
| Create Project Budget | 2024-10-28 | 2024-10-28 | Marrec,Nic |
| Budgeting | 2024-10-28 | 2024-10-28 | Marrec,Nic |
| Gantt Chart Creation | 2024-10-28 | 2024-10-28 | Karim |
| Week 1 Meeting Agenda | 2024-10-28 | 2024-10-28 | William |
| Initial meeting | 2024-10-29 | 2024-10-29 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Brainstorming Design Ideas | 2024-10-29 | 2024-10-29 | Marrec,William,Garrett,Nic |
| Team Contract | 2024-10-29 | 2024-10-29 | Katrina |
| Brainstorming Documentation | 2024-10-30 | 2024-10-30 | Nic |
| Week 1 Meeting Minutes | 2024-10-29 | 2024-10-29 | William |
| Requirement Writing | 2024-10-30 | 2024-10-30 | William |
| Requirement Review | 2024-10-31 | 2024-10-31 | Katrina |
| Outline Project Contraints | 2024-10-31 | 2024-10-31 | Katrina |
| Gantt Chart Review | 2024-10-31 | 2024-10-31 | Garrett |
| Model Mobile Base with Color Sensors | 2024-11-01 | 2024-11-01 | Nic,Karim |
| Basic Spiral Traversal Code do Robot (Ignore obstacles) | 2024-11-02 | 2024-11-02 | Marrec |
| Test Traversal Code (Ensure entire surface is covered) | 2024-11-03 | 2024-11-03 | Katrina |
| Test Ultrasonic Sensors for Positioning Accuracy | 2024-11-03 | 2024-11-03 | Katrina |
| Week 2: First Semi-Functional Product is Built | 2024-11-04 | 2024-11-11 | |
| Update Timesheets | 2024-11-04 | 2024-11-11 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Weekly Progress Meeting | 2024-11-04 | 2024-11-04 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Initial Design Review / Evaluation | 2024-11-04 | 2024-11-04 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Model Cube "Pickup" Mechanism | 2024-11-05 | 2024-11-05 | Nic,Karim |
| Update hardware sensor placement | 2024-11-05 | 2024-11-05 | Nic |
| Increase hardware carrying capacity | 2024-11-06 | 2024-11-06 | Nic |
| Model raising and lowering of the ultrasonic sensor | 2024-11-05 | 2024-11-05 | Nic |
| Implement mobile base | 2024-11-06 | 2024-11-06 | Nic,Karim |
| Implement Color Classification Algorithm | 2024-11-05 | 2024-11-05 | William |
| Write Code for Lowering and Raising of US Sensor | 2024-11-07 | 2024-11-07 | Marrec,Garrett |
| Implement raising and lowering of the ultrasonic sensor | 2024-11-07 | 2024-11-07 | Nic,Karim |
| Add Water Detection to Traversal Code | 2024-11-06 | 2024-11-06 | Marrec,William |
| Add Water Avoidance to Traversal Code | 2024-11-07 | 2024-11-07 | Marrec,William,Garrett |
| Test Water Avoidance (Cover all green while avoiding water) | 2024-11-08 | 2024-11-08 | Karim,Katrina |
| Week 2 Meeting Agenda | 2024-11-08 | 2024-11-08 | William |
| Week 2 Meeting Minutes | 2024-11-08 | 2024-11-08 | William |
| Update Budget | 2024-11-09 | 2024-11-09 | Karim |
| Hardware Design Document | 2024-11-09 | 2024-11-09 | Nic |
| Software Design Document | 2024-11-09 | 2024-11-09 | Marrec,William,Garrett |
| Model/System Description | 2024-11-10 | 2024-11-10 | Marrec,William |
| Test Color Sensors to Measure Floor Colour at Cube Height | 2024-11-10 | 2024-11-10 | Katrina |
| Test Color sensors for cubes | 2024-11-10 | 2024-11-10 | Karim,Katrina |
| Initial Integration Test for Sensors and Movement | 2024-11-11 | 2024-11-11 | Karim,Katrina |
| Week 2 Full Testing Documentation | 2024-11-11 | 2024-11-11 | Karim,Katrina |
| Report Test Results to Software Team | 2024-11-11 | 2024-11-11 | Marrec,William,Garrett |
| Week 3: Prototype is Finalized | 2024-11-12 | 2024-11-18 | |
| Test Color sensors for cubes | 2024-11-12 | 2024-11-12 | Katrina |
| Test Color Sensors to Measure Floor Colour at Cube Height | 2024-11-12 | 2024-11-12 | Katrina |
| Create Design Review Presentation Draft | 2024-11-12 | 2024-11-12 | Nic,Karim |
| Weekly Progress Meeting | 2024-11-12 | 2024-11-12 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Initial Integration Test for Sensors and Movement | 2024-11-13 | 2024-11-13 | Karim,Katrina |
| Update Design Review Presentation based on feedback | 2024-11-13 | 2024-11-13 | Nic |
| Discuss Current Problems in Design and Potential Solutions | 2024-11-12 | 2024-11-12 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Implement Full Navigation Algorithm For Robot | 2024-11-13 | 2024-11-13 | Marrec,William |
| Test Navigation System (Percentage of poop collected) | 2024-11-14 | 2024-11-14 | Karim,Katrina |
| Week 3 Meeting Agenda | 2024-11-14 | 2024-11-14 | William |
| Week 3 Meeting Minutes | 2024-11-14 | 2024-11-14 | William |
| Full Initial Hardware Integration Test | 2024-11-15 | 2024-11-16 | Nic,Karim |
| Full Initial Software Integration Test | 2024-11-15 | 2024-11-16 | Karim,Katrina |
| Create Project Report Template | 2024-11-15 | 2024-11-17 | Garrett |
| Reflect on Integration Tests / Plan Upcoming Tests | 2024-11-17 | 2024-11-17 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Bug Fixes on Overall Code Design | 2024-11-17 | 2024-11-17 | Marrec,William,Garrett |
| Implement Cube "Pickup" Mechanism | 2024-11-17 | 2024-11-17 | Nic,Karim |
| Create Project Report Template | 2024-11-17 | 2024-11-17 | Nic |
| Arrange cable management | 2024-11-18 | 2024-11-18 | Nic |
| Implement Cube Detection and Approach Algorithm | 2024-11-18 | 2024-11-18 | Marrec |
| Week 4: Report Draft and Integration Testing is Complete | 2024-11-19 | 2024-11-28 | |
| Final Integration Testing (Full system HW and HW) | 2024-11-19 | 2024-11-19 | Karim,Katrina |
| Centralize Everyone's Contributions into the Report | 2024-11-28 | 2024-11-28 | Garrett |
| Team Design Reviews | 2024-11-28 | 2024-11-28 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Bug Fixes In Cube Pickup | 2024-11-28 | 2024-11-29 | Marrec,Garrett |
| Cube Avoidance algorithm | 2024-11-21 | 2024-11-21 | Marrec |
| Write Return to Track Algorithm | 2024-11-21 | 2024-11-21 | William |
| Update water avoidance to avoid all water | 2024-11-22 | 2024-11-22 | William,Nic |
| Revamp Navigation Algorithm | 2024-11-22 | 2024-11-22 | Marrec,Garrett |
| Optimize Color Classification Algorithm | 2024-11-22 | 2024-11-22 | William |
| Test Revamped Navigation Algorithm | 2024-11-23 | 2024-11-23 | Karim,Katrina |
| Final Hardware Optimizations | 2024-11-24 | 2024-11-24 | Nic |
| Work Buffer | 2024-11-25 | 2024-11-28 | |
| Week 5 | 2024-11-27 | 2024-12-03 | |
| Final Complete Integration Test | 2024-11-27 | 2024-12-01 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Finalize Budget | 2024-11-27 | 2024-11-27 | Garrett |
| Final HW Design Documentation | 2024-11-27 | 2024-11-27 | Nic,Karim |
| Final SW Design Documentation | 2024-11-27 | 2024-11-27 | Garrett |
| Final Bug Fixes | 2024-11-27 | 2024-11-27 | Marrec,William,Garrett |
| Final Testing Documentation | 2024-11-28 | 2024-11-28 | Katrina |
| Team Strategy Meeting | 2024-11-28 | 2024-11-28 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Finalize Gantt Chart | 2024-11-28 | 2024-11-28 | Karim |
| Organize and Compile Test Data into report | 2024-11-28 | 2024-11-29 | Karim,Katrina |
| Project Management Section of Report | 2024-11-29 | 2024-11-29 | Marrec |
| Add Software Design and Iterations Into report | 2024-11-30 | 2024-11-30 | William |
| Add Hardware Design and Iterations into report | 2024-11-30 | 2024-11-30 | Nic |
| Create Professional Sketches of Robot for Report | 2024-11-30 | 2024-11-30 | Karim |
| Final test runs for edge cases (low battery, etc.) | 2024-11-30 | 2024-11-30 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Prepare for Demo | 2024-11-29 | 2024-12-01 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Final Report review | 2024-12-01 | 2024-12-01 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Work Buffer | 2024-12-02 | 2024-12-03 | Marrec,William,Garrett,Nic,Karim,Katrina |
| Demo | 2024-12-04 | 2024-12-04 | Marrec,William,Garrett,Nic,Karim,Katrina |

Figure A.3.i

| | Hours Per Person: Marrec | Will | Garrett | Nic | Karim | Katrina | Total Hours |
|---|---|---|---|---|---|---|---|
| Total For Each Person: | 45 | 45 | 45 | 45 | 45 | 44.5 | 269.5 |
| **Milestone/Week 1: Intial Plan Created** | | | | | | | |
| Initial Meeting | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 9 |
| Brainstorming Design Ideas | 2.5 | 2.5 | 2.5 | 2.5 | | | 10 |
| Budgeting | 1.5 | | | 1.5 | | | 3 |
| Brainstorming Documentation | | | | 2 | | | 2 |
| Gantt Chart Creation | | | | | 4 | | 4 |
| Team Contract | | | | | | 1 | 1 |
| Model Mobile Base With Color Sensors | | | | 2 | 1 | | 3 |
| Week 1 Meeting Agenda | | 1 | | | | | 1 |
| Week 1 Meeting Minutes | | 0.5 | | | | | 0.5 |
| Requirement Writing | | 2.5 | | | | | 2.5 |
| Requirement Review | | | | | | 1.5 | 1.5 |
| Outline Project Constraints | | | | | | | 0 |
| Budget Documentation | 1 | | | | | | 1 |
| Gantt Chart Review | | | 2 | | | | 2 |
| Basic Spiral Traversal Code for Robot (Ignore obstacles) | 2 | | | | | | 2 |
| Test Traversal Code (Ensure entire surface is covered) | | | | | | 1.5 | 1.5 |
| Test Ultrasonic Sensors for Positioning Accuracy | | | | | | 1 | 1 |
| Update Timesheet | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 1.5 |
| | | | | | | | 0 |
| | | | | | | | 0 |
| **Milestone/Week 2: First Semi-Functional Product is Built** | | | | | | | 0 |
| Weekly Progress Meeting | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| Initial Design Review / Evaluation | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 3 |
| Model Cube "Pickup" Mechanism | | | | 1 | 1 | | 2 |
| Model raising and lowering of the ultrasonic sensor | | | | 1 | | | 1 |
| Implement mobile base | | | | 2 | 0.5 | | 2.5 |
| Implement Cube "Pickup" Mechanism | | | | 1 | 1 | | 2 |
| Implement raising and lowering of the ultrasonic sensor hardware | | | | 1 | 1 | | 2 |
| Add Water Consideration to Traversal Code | 2.5 | 1 | | | | | 3.5 |
| Test Water Avoidance (Cover all green while avoiding water) | | | | | 2 | 2 | 4 |
| Week 2 Meeting Agenda | | 1 | | | | | 1 |
| Week 2 Meeting Minutes | | 0.5 | | | | | 0.5 |
| Update Timesheets | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 1.5 |
| Finalize Budget | | | | | 1.5 | | 1.5 |
| Hardware Design Document | | | | 3 | | | 3 |
| Software Design Document | 1.5 | 1 | 1.5 | | | | 4 |

Figure A.4.a Final Project Budget

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Model/System Description | 1 | 2 | | | | | 3 |
| Week 2 Full Testing Documentation | | | | | 3 | 3 | 6 |
| Report Test Results to Software Team | 1 | 1 | 1 | | | 1 | 4 |
| | | | | | | | 0 |
| | | | | | | | 0 |
| **Milestone/Week 3: Prototype is Finalized and Optimization has begun** | | | | | | | 0 |
| Test Color sensors for cubes | | | | | | 2 | 2 |
| Test Color Sensors to Measure Floor Colour at Cube Height | | | | | | 2 | 2 |
| Write Code for Lowering and Raising of US Sensor | 1.5 | | 1.5 | | | | 3 |
| Initial Integration Test for Sensors and Movement | | | | | 1.5 | 1.5 | 3 |
| Weekly Progress Meeting | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| Design Review Presentation Create | | | 3 | | 0.5 | | 3.5 |
| Discuss Current Problems in Design and Potential Solutions | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 3 |
| Implement Full Navigation Algorithm For Robot | 4 | 4 | | | | | 8 |
| Test Navigation System (Percentage of poop collected) | | | | | 1.5 | 2 | 3.5 |
| Full Initial Hardware Integration Test | | | | 2 | 1 | | 3 |
| Full Initial Software Integration Test | 1 | | | | 1 | 1 | 3 |
| Reflect on Integration Tests / Plan Upcoming Tests | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 3 |
| Bug Fixes on Overall Code Design (Unsure where yet) | 2.5 | 2.5 | 2.5 | | | | 7.5 |
| Week 3 Meeting Agenda | | 1 | | | | | 1 |
| Week 3 Meeting Minutes | | 0.5 | | | | | 0.5 |
| Arrange cable management | | | | 1 | | | 1 |
| Create Project Report Template | | | 1.5 | | | | 1.5 |
| | | | | | | | 0 |
| **Milestone/Week 4: Report Draft is Complete and Robot is Demo Ready** | | | | | | | 0 |
| Final Integration Testing (Full system SW and HW) | | | | | 2 | 2 | 4 |
| Centralize Everyone's Contributions into the Report | | | 3 | | | | 3 |
| Team Design Reviews (Nov 20) | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 9 |
| Final HW Design Documentation | | | | 3 | 2 | | 5 |
| Final SW Design Documentation | | | 3 | | | | 3 |
| Final Testing Documentation | | | | | | 3 | 3 |
| Final Bug Fixes | 3 | 3 | 3 | | | | 9 |
| Prepare for Demo | 2 | 2 | 2 | 2 | 2 | 2 | 12 |
| Report Buffer | 2 | 3 | 2 | 3 | 3 | 2 | 15 |
| | | | | | | | 0 |
| | | | | | | | |
| **Milestone/Week 5: Report is Finalized and Demo is performed** | | | | | | | 0 |
| Final Report review | 2 | 2 | 2.5 | 2 | 2 | 2 | 12.5 |
| Work Buffer | 4 | 4 | 4 | 5 | 3.5 | 4 | 24.5 |
| Final test runs for edge cases (low battery, etc.) | 2 | 2 | 2 | 2 | 2 | 2 | 12 |

Figure A.4.b Final Project Budget

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Demo | 1 | 1 | 1 | 1 | 1 | 1 | 6 |

Figure A.4.c Final Project Budget

# B. Specifications

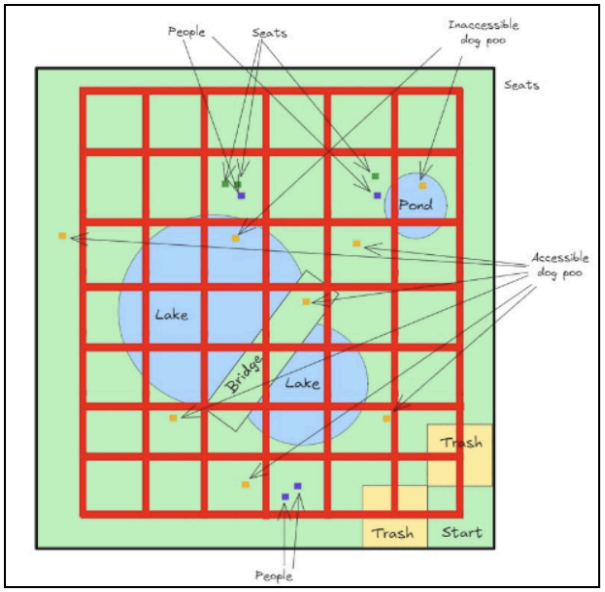



Figure B.1. Schematic of the prototype trial map

| Component | Measurement (cm) |
|---|---|
| Outer Wheel Base | 12.5 |
| Inner Wheel Base | 8.2 |
| Total Robot Width | 17.5 |
| Metal Ball to Metal Ball Distance | 9.5 |
| Total Robot Height | 13.5 |
| Side US Sensor Height | 9.5 |
| Front US Sensor Height at Top Position | 7 |
| Front US Sensor Height at Bottom Position | 1.2 |
| Inside Cage Width | 6.2 |
| Total Robot Length | 24.5 |
| Left Color Sensor to Right Color Sensor | 14 |
| Right Color Sensor Height | 2.7 |

Table B.1. Total dimensions of the robot

# C. Design

## Doc C.1: Build 1 - Dual Brick Robot (Not Chosen) Full Specification Document

**Structure:** A two-brick setup, with each brick stacked one on top of the other. The base is designed as a cage to drag cubes, supported by two ball bearings at the front and wheels at the back for movement.

**Components:**

- **Bricks:** 2 (Brick 1 on top, Brick 2 below)
- **Motors:** 2 motors, each driving a wheel for movement and steering.

**Sensor Placement and Purpose:**

- **Front-Facing US Sensor:** Positioned close to the ground, facing forward. Used to detect obstacles (cubes) directly in front of the robot, aiding in cube collection.
- **Right-Facing US Sensor:** Positioned to face right, enabling wall-following functionality by maintaining a specified distance from the wall.
- **Front-Left Color Sensor:** Placed at the front left of the robot, primarily for navigation. This sensor detects water to the left of the robot. The navigation plan is to circle the field in a counterclockwise path with gradual left turns, mapping and "remembering" water areas to avoid them later. When encountering water, the robot uses the detected blue color as a boundary to circle the area.
- **Forward-Facing Color Sensor:** Positioned near the front-facing US sensor, facing forward. This sensor is intended for block color detection, assisting in identifying and categorizing blocks (e.g., by color) as the robot encounters them.
- **Gyro Sensor:** Positioned centrally to measure rotational orientation. It enables the robot to determine when it's parallel to a wall, aiding in wall-following and navigation accuracy by measuring turns and aligning with the field's boundaries.

**Navigation Strategy:**

- **Field Sweep:** The robot plans to conduct a comprehensive sweep of the field by moving in a circular pattern with left turns, gradually spiraling inward.
- **Water Mapping:** When the front-left color sensor detects water on the left side, the robot circles the detected area, uses the water's edge as a guide, and maps the location for future avoidance.
- **Wall Following:** Using the right-facing US sensor and gyro sensor, the robot maintains a set distance from walls, helping it navigate along the boundaries of the field.
- **Cube Detection:** The forward-facing US sensor detects cubes directly in front of the robot, allowing for targeted movement towards blocks.
- **Block Color Identification:** The forward-facing color sensor distinguishes block colors for sorting or specific handling.

Doc C.2: Build 2 - Arm-Based Robot (Not Chosen) Full Specification Document

**Structure:** Arm with Color Sensor - a two-axis arm (moving along the X and Y axes) is equipped with a color sensor. The arm extends over each square to probe for cubes, allowing for targeted cube collection without needing to drive directly over every area.
**Square-by-Square Navigation:** This implementation follows a systematic grid-based approach, advancing square by square across the field.

**Components:**

- **Motors:** 4 in total – 2 for driving, 2 for controlling the arm's X and Y axes.
- **Sensors:** 4 in total – 1 Ultrasonic (US) sensor, 2 color sensors, and 1 Gyro sensor.

**Sensor Placement and Functionality:**

- **Ultrasonic (US) Sensor:** Positioned to assess the presence of cubes in each square. Before driving into a square, the robot does a 45-degree sweep (turning slightly left and right) to scan the area. If any object (cube or obstacle) is detected within range, the robot assesses whether to proceed or avoid.

- **Left Front Color Sensor:** Positioned on the left front side of the robot, primarily for detecting water areas. This sensor is essential for avoiding water hazards, prompting the robot to navigate around any detected water regions instead of driving through them.
- **Arm-Mounted Color Sensor:** Mounted on an arm that can extend over each square to probe for cubes. When a cube is detected by the US sensor in a square, the arm's color sensor checks the cube's color. If the cube is of interest, the robot collects it by driving over it and using the cage-like system for containment.
- **Gyro Sensor:** Provides orientation feedback, allowing the robot to track its direction on the grid (north, south, east, or west). This helps maintain a precise grid-based navigation and ensures the robot moves in a controlled and predictable pattern.

**Grid-Based Navigation:**

- **Square-by-Square Progression:** The robot follows a structured grid layout, moving from one square to the next systematically, ensuring thorough coverage of the entire field.
- **Obstacle and Cube Detection:** At each square, the US sensor performs a 45-degree sweep to detect any obstacles or cubes. If a cube is present, the arm with the color sensor extends to check its color. If the cube is desired, the robot collects it by driving over it, utilizing the under-robot cage system.
- **Water Avoidance:** When the front-left color sensor detects water, the robot adjusts its path to avoid entering the hazardous square.
- **Grid Line Navigation:** The color sensors help the robot align with grid lines for precise positioning and navigation, maintaining a controlled grid sweep.

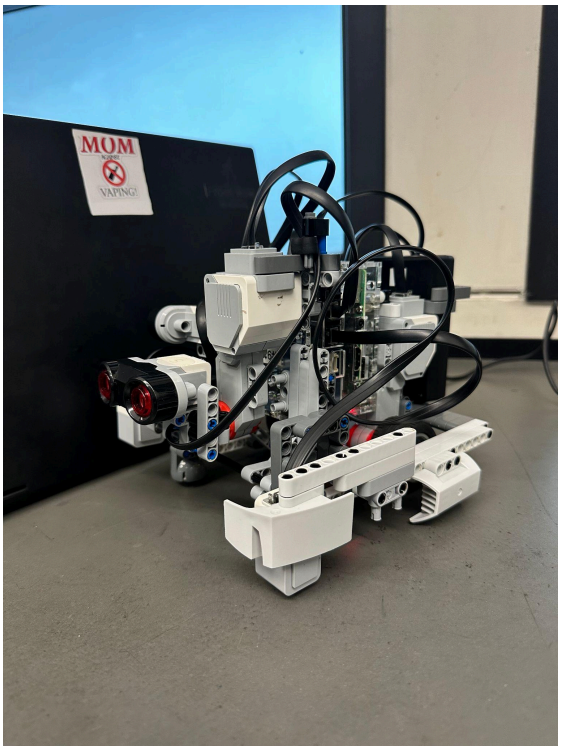



Image C.1. Final Robot Design Left Angle

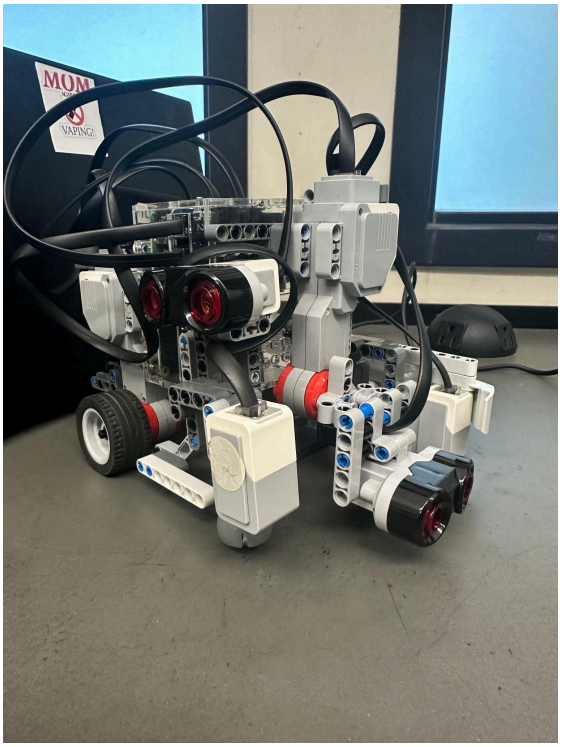



Image C.2 Final Robot Design Right Angle

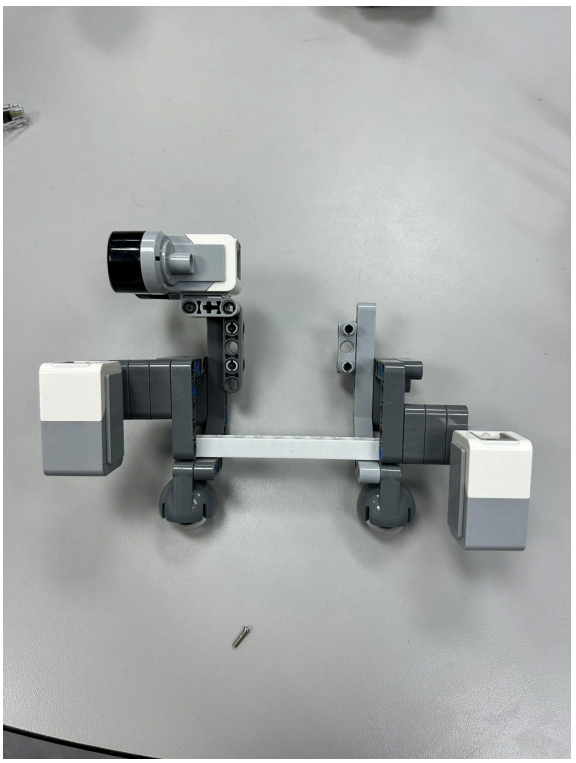


Image C.3. Second Iteration of the Front Sensor Frame Module


Image C.4. Drive Train Module

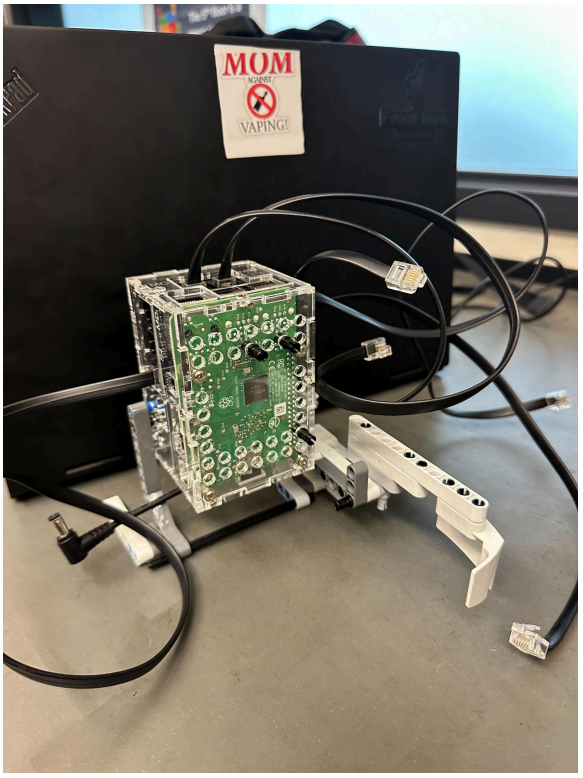



Image C.5. Core Chassis & BrickPi Module

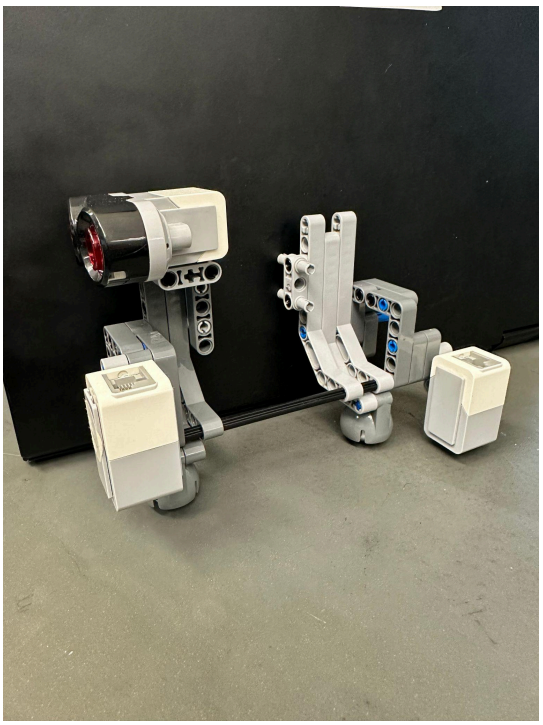


Image C.6. Front Sensor Frame Module

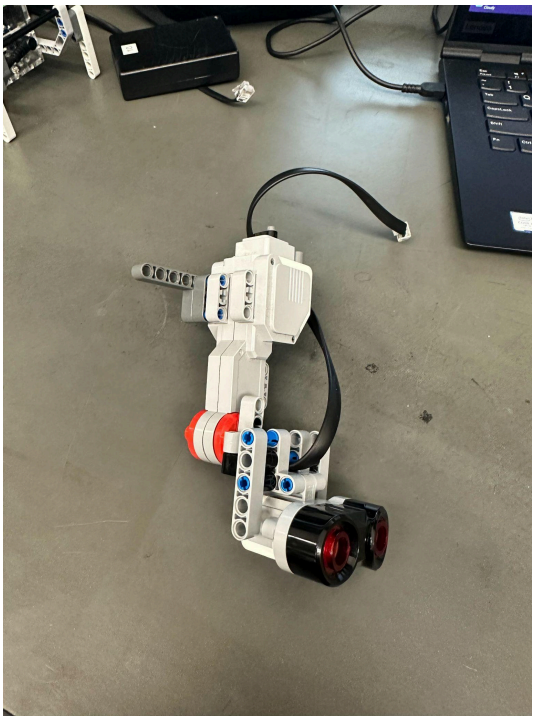


Image C.7. Front US Sensor Module

Image C.8. Front US Sensor Module Cable Hold

Image C.9. Cable Bend Behind Front US Sensor

Image C.10 Top Of Front US Sensor Cable View

Image C.11. Front US Sensor Full Cable view

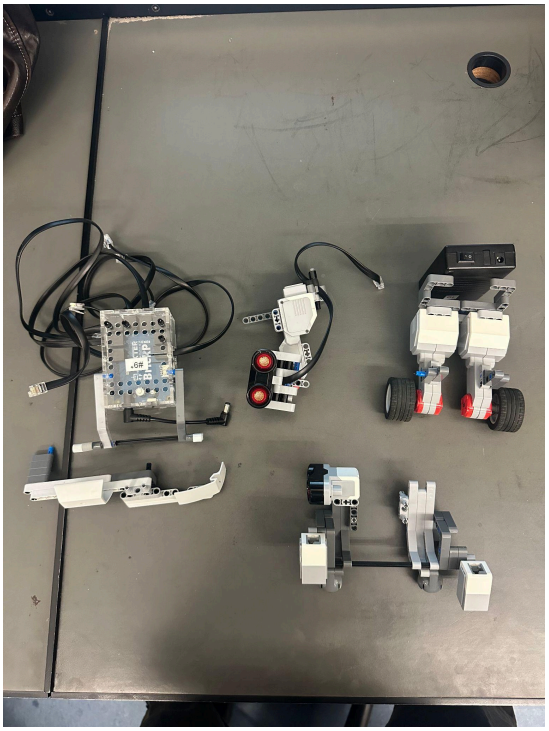


Image C.12. All Modules Exploded View

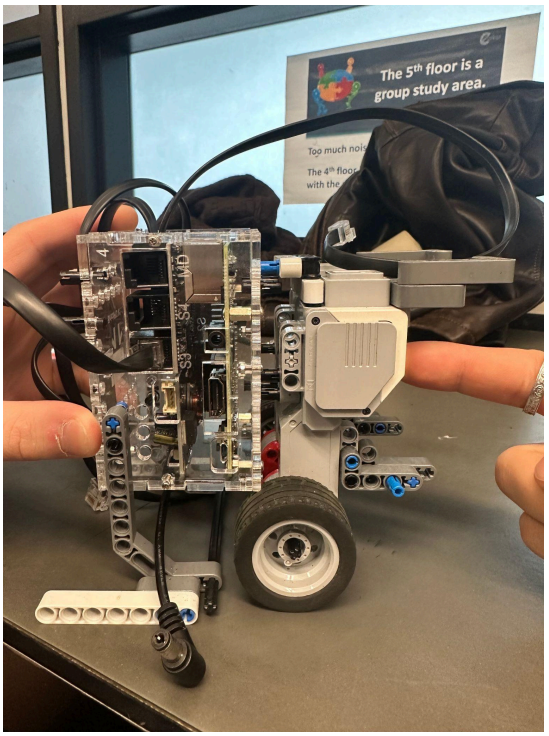



Image C.13. Drivetrain and Chassis Attachment View

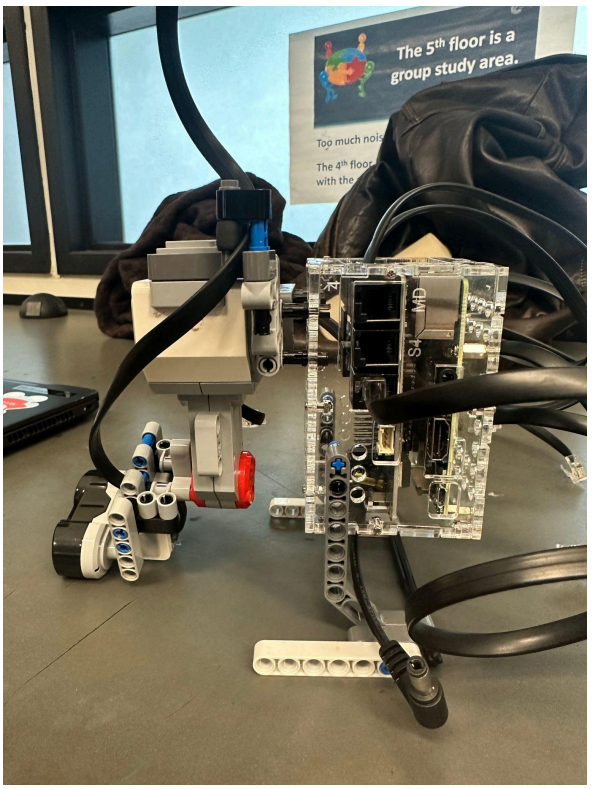
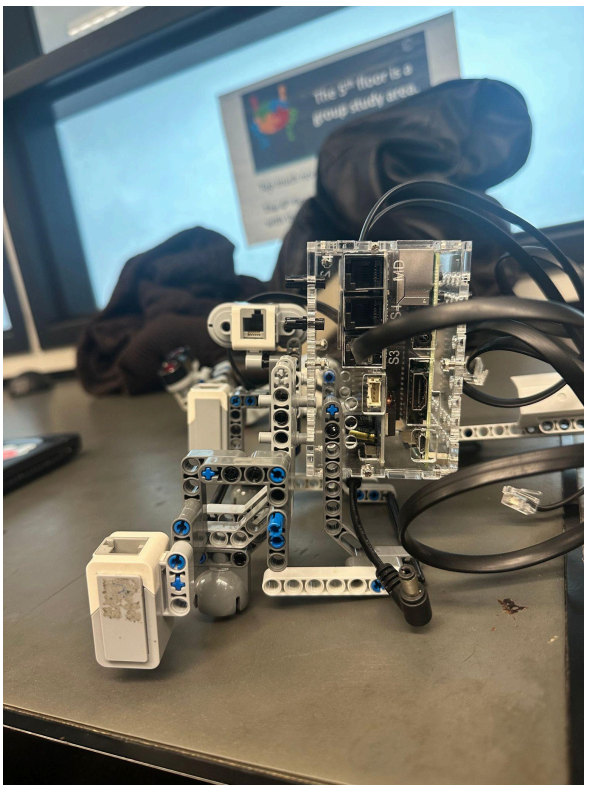



Image C.14. Front US Sensor and Chassis Attachment



C.15. Sensor Frame Module and Chassis Attachment

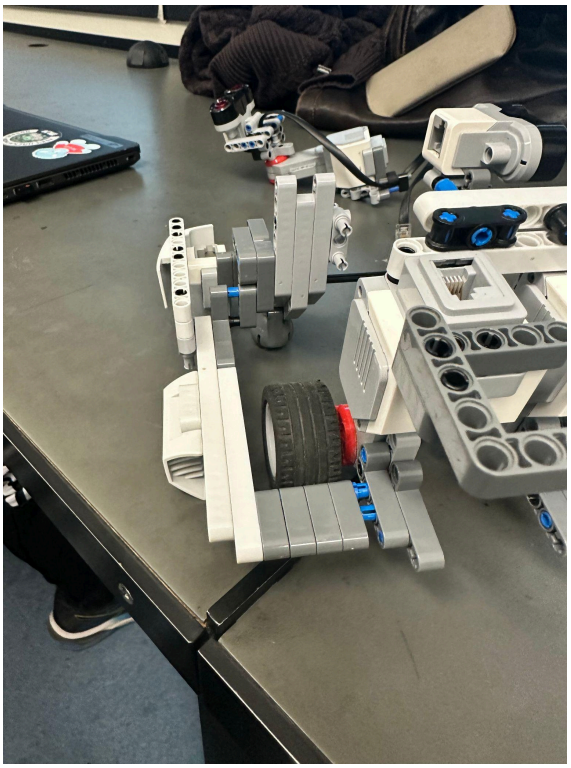
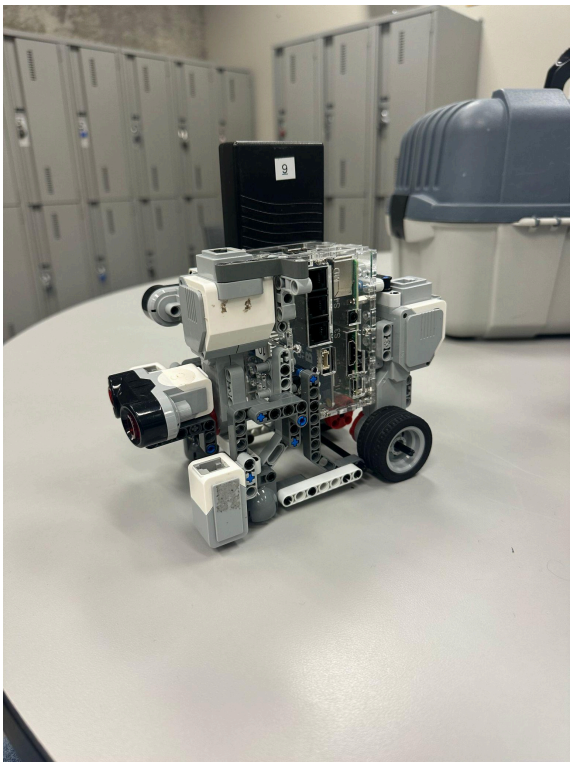


Image C.16. Wall Guard attachment to Drivetrain and Front Sensor Frame



C.17. First Design of Robot

Table C.1: CSR Color Model Data

| File | Average | Standard Deviation | Normalized RGB |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **CSR_green_block** | (38.7, 169.1, 55.7) | (0.90, 0.73, 0.53) | (0.15, 0.64, 0.21) |
| **CSR_ground** | (14, 22, 4.9) | (0.94, 0.96, 0.57) | (0.34, 0.54, 0.12) |
| **CSR_orange_block** | (351.9, 84.2, 51.4) | (1.78, 1.79, 1.06) | (0.72, 0.17, 0.11) |
| **CSR_purple_block** | (83.8, 58.6, 75.9) | (1.03, 0.78, 0.74) | (0.38, 0.27, 0.35) |
| **CSR_red_tape** | (29.9, 8.7, 4.6) | (0.83, 0.98, 0.59) | (0.69, 0.20, 0.11) |
| **CSR_water** | (4.5, 6.3, 14.3) | (0.88, 1.08, 0.75) | (0.18, 0.25, 0.57) |
| **CSR_yellow_block** | (354.9, 231.7, 29.8) | (1.02, 1.25, 0.86) | (0.58, 0.38, 0.05) |
| **CSR_trash** | (48.04, 33.58, 7.71) | (1.08, 1.02, 0.91) | (0.54, 0.38, 0.09) |

Figure C.1: CSR Normalized RGB Color Space Model

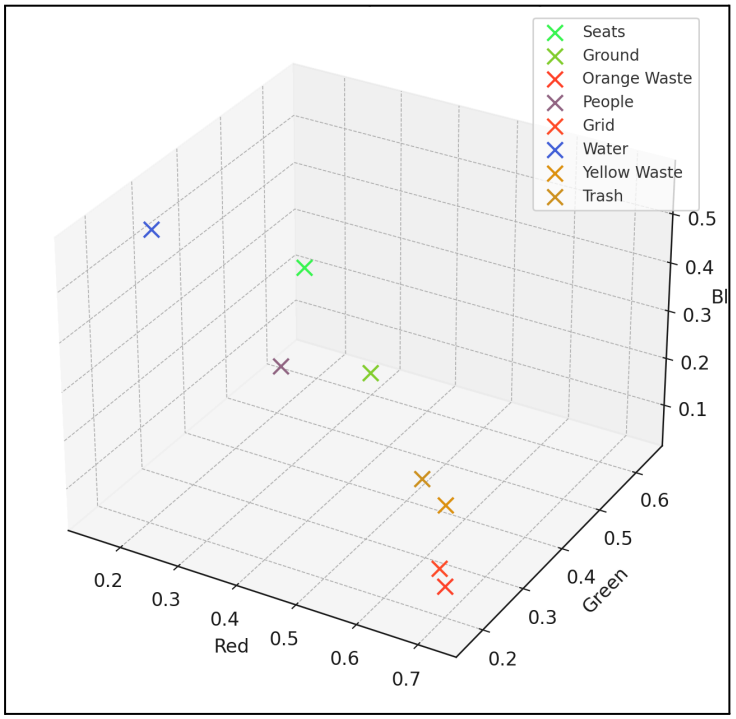



Figure C.2: CSL Normalized RGB Color Space Model

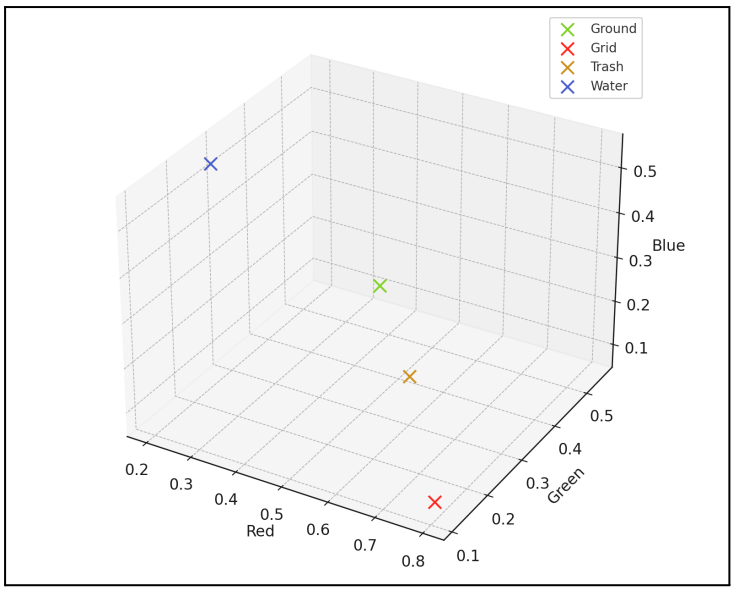



Table C2: CSL Color Model Data

| File | Average | Standard Deviation | Normalized RGB |
|---|---|---|---|
| **CSL_ground** | (92.5, 150.3, 30.9) | (0.85, 1.28, 0.87) | (0.34, 0.55, 0.11) |
| **CSL_red_tape** | (223.5, 31.5, 25.7) | (1.15, 1.00, 0.94) | (0.80, 0.11, 0.09) |
| **CSL_trash** | (322.4, 224.5, 45.0) | (0.95, 0.96, 0.96) | (0.54, 0.38, 0.08) |
| **CSL_water** | (31.0, 39.2, 83.3) | (0.96, 0.87, 0.79) | (0.20, 0.26, 0.54) |

Figure C.3 Color Classifier Dataflow

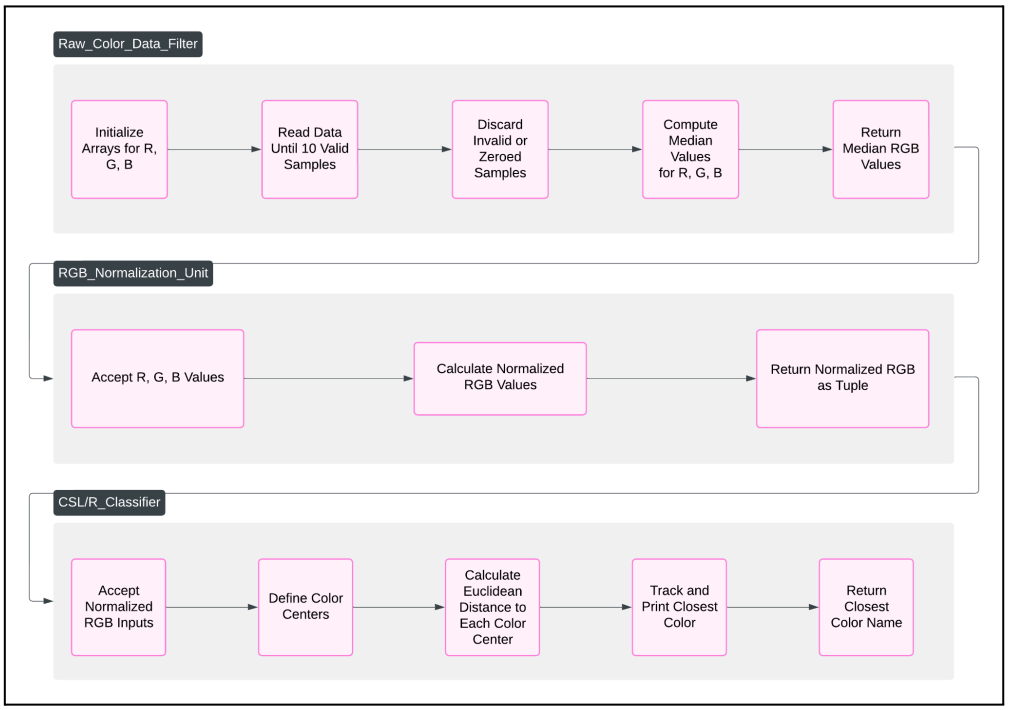


Table C.3 Software Modules Overview

| **Module** | **Description** | **Utilized Algorithms** |
|---|---|---|
| Robot Navigation | Responsible for the robot's traversal of the environment. The module should allow the robot to travel the majority of the course and simultaneously evade water. | Base Traversal Algorithm, Water Avoidance Algorithm, Single Spiral with Spanning Algorithm, Colour Classification Algorithm |
| Block Detection and Response | Responsible for the robot's ability to detect and respond to cubes. The module should allow the robot to detect cubes in front of it, safely approach them, scan their colour, pick them up if they are poop or avoid them if they are obstacles. | Cube Detection Algorithm, Classification, Capture, and Avoidance Algorithm, Colour Classification Algorithm |
| Integration | Integrate both modules to allow the robot to traverse the map while avoiding water while simultaneously collecting poop cubes and avoiding obstacle cubes | Integration Algorithm |

Table C.4: Motors and Sensor utilized in Cube Detection and Approach algorithm

| **Motor/Sensor** | **Description** |
|---|---|
| LML (Large Motor Left) | Large Motor mounted on the back left of the robot providing power to left wheel |
| LMR (Large Motor Right) | Large Motor mounted on the back right of the robot providing power to right wheel |
| LMF (Large Motor Front) | Large Motor that rotates the USF after an object is detected to determine if detected object is cube or wall |
| USF (Ultrasonic Forward) | Detects distance to objects positioned in front of the robot. |

Table C.5: Detailed Iterations of Cube Detection and Approach Algorithm

| **Iteration Number** | **Description** |
|---|---|
| V1.0 | The USF is able to detect a block placed directly in front of the robot. After which point the robot rotates first to the left and then to the right to determine the exact position of the cube it has discovered. At the end of this process, the robot is facing directly at the detected cube but cannot yet approach. |
| V1.1 | Following the Block Detection process implemented in V1.0, the robot is now able to approach the cube directly until it senses the cube is less than 3 cm from the robots front and is now ready to execute the Classification, Capture and Avoidance Algorithm. The robot still does not distinguish between cubes and walls. |
| V1.2 | The robot successfully implements the features described in V1.0, V1.1 but can now distinguish between a detected object and detected block. After first detecting an object with the USF, the LMF rotates the USF and compares the higher position reading to the low position reading to determine if the object is a cube. |

Table C.6: Motors and Sensor utilized in Cube Classification, Capture and Avoidance algorithm

| **Motor/Sensor** | **Description** |
|---|---|
| LML (Large Motor Left) | Large Motor mounted on the back left of the robot providing power to left wheel |
| LMR (Large Motor Right) | Large Motor mounted on the back right of the robot providing power to right wheel |
| LMF (Large Motor Front) | Large Motor that rotates the USF after an object is detected to determine if detected object is cube or wall |
| CSR (Colour Sensor Right) | Right colour sensor is used to determine colour of cube directly beneath it and classify it as poop or obstacle |
| USF (Ultrasonic Forward) | Detects distance to objects positioned in front of the robot. |

Table C.7: Detailed Iterations of Cube Classification, Capture and Avoidance, Algorithm

| **Iteration Number** | **Description** |
|---|---|
| V2.0 | Upon the completion of the Block Detection and Approach Algorithm, the robot can turn slightly to the left and drive forward until the CSR is directly above the cube it is trying to classify. Still, the robot often drives past the cube preventing the CSR from getting an accurate reading. The colour classification algorithm is not yet called. |
| V2.1 | The robot's turning and approach were tweaked and fine tuned so that it can reach cubes more accurately and consistently. It is now able to position its CSR directly above the detected block almost every time. The colour classification algorithm is not yet called. |
| V2.2 | After utilizing the turning and approach parameters described in V2.1 to position the CSR directly above the detected cube, the robot can now classify the cubes colour using the Colour Classification Algorithm. The cubes colour is printed to the Standard Output of the program. |
| V2.3 | The robot successfully implements all features described up to and including V2.2 and in addition can respond accurately when a poop is detected. The robot first raises the USF, rotates slightly to the left and drives forward to collect the cube in the robot's undermounted cage. |
| V2.4 | The robot successfully implements all features described up to and including V2.3 and in addition can now respond accurately when an obstacle is detected. After detecting an obstacle, the robot successfully backs up slightly to avoid the cube. A more advanced avoidance algorithm is planned for when the Avoidance algorithm is integrated with the navigation algorithm |

Table C.8.: Legend for Base Traversal Visualization

| **Symbol** | **Meaning** |
|---|---|

| | |
|---|---|
| Black Arrow | Indicates robot's movement path and direction |
| Purple + | Point of increase in forward (USF) distance threshold |
| Red X | Point of increase in side (USS) distance threshold |
| Purple Arrow | Polling direction of USF |
| Red Arrow | Polling direction of USS |

Table C.9: Motors and Sensor utilized in Base Traversal algorithm

| **Motor/Sensor** | **Description** |
|---|---|
| LML (Large Motor Left) | Large Motor mounted on the back left of the robot providing power to left wheel |
| LMR (Large Motor Right) | Large Motor mounted on the back right of the robot providing power to right wheel |
| USF (Ultrasonic Forward) | Polls forward facing wall until below left turn threshold |
| USS (Ultrasonic Side) | Polls side facing wall to maintain side distance target |

Table C.10: Detailed Iterations of Base Traversal Algorithm

| **Iteration Number** | **Description** |
|---|---|
| V1.0 | The robot continuously polls the forward facing wall as it drives forward. When the reading from front USF is less than a specified distance threshold, the robot completes a left turn. After the robot has completed three rotations, the distance threshold is increased so that upon reaching the next wall, the robot turns earlier allowing it to complete a spiral inside of the one it just completed. It was noticed that over time the robot would drift towards or away from the wall causing its spirals to become less and less steady over time. |
| V1.1 | Polling of the adjacent wall using the USS side was implemented so the robot could travel at a set distance from the side wall. If the robot began drifting away from the side wall the left wheel speed was increased. If the robot began drifting towards the wall, the right wheel speed was increased. After every fourth turn the robot made the side distance target was increased. This update allowed the robot to traverse the entirety of the map using successive spirals. |

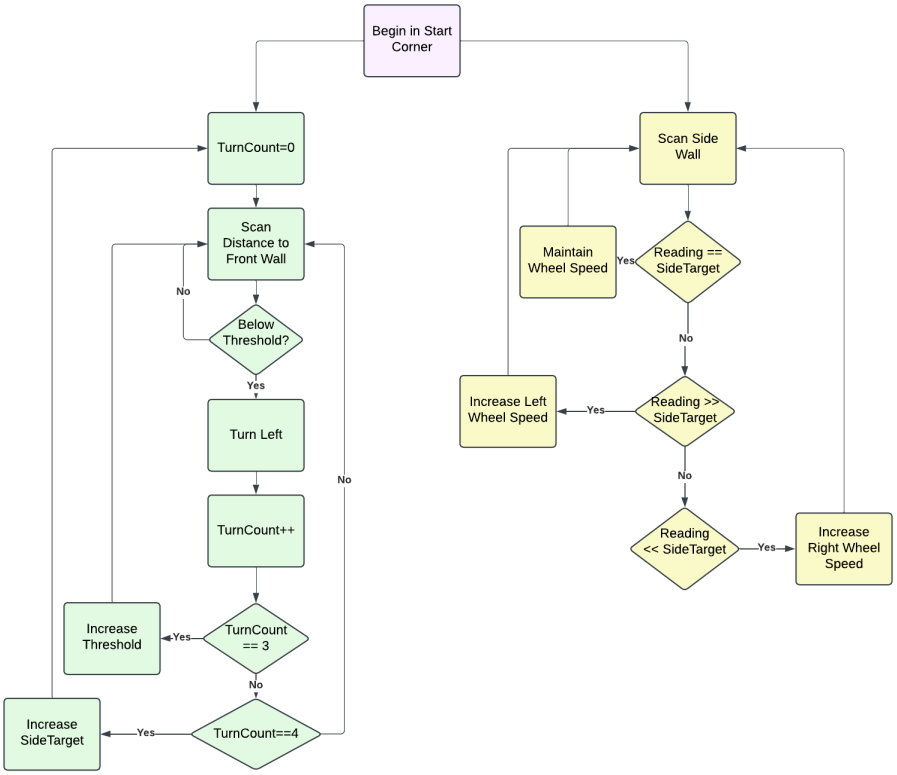



Figure C.4: Flowchart of Base Traversal Algorithm

Table C.11: Legend for Water Avoidance visualization

| **Symbol** | **Meaning** |
|---|---|
| Black Arrow | Indicates robot's main movement path and direction |
| Blue Arrow | Indicates robot's water avoidance phase |
| Pink Arrow | Bridge detected |
| Purple + | Point of increase in forward (USF) distance threshold |
| Red X | Point of increase in side (USS) distance threshold |

Table C.12: Motors and Sensor utilized in Water Avoidance algorithm

| **Motor/Sensor** | **Description** |
|---|---|
| LML (Large Motor Left) | Large Motor mounted on the back left of the robot providing power to left wheel |
| LMR (Large Motor Right) | Large Motor mounted on the back right of the robot providing power to right wheel |
| USF (Ultrasonic Forward) | Polls forward facing wall until below left turn threshold |

| | |
|---|---|
| USS (Ultrasonic Side) | Polls side facing wall to maintain side distance target |
| CSL (Color Sensor Left) | Polls the ground on the left side of the robot to detect water |
| CSR (Color Sensor Right) | Polls the ground on the right side of the robot to detect water |

Table C.13: Detailed Iterations of Water Avoidance Algorithm

| **Iteration Number** | **Description** |
|---|---|
| V2.0 | The robot begins travelling forward following a path planned by the base traversal algorithm. As the robot travels forward, the ground is continuously polled using the CSR and CSL. The readings are passed into the colour classification algorithm to determine if the ground below the sensor is water. If water is detected by either sensor the message “CSX: Detects Water” (where X corresponds to the colour sensor that detects water) is printed to Standard Out. The robot does not make any attempt to evade water that it detects. |
| V2.1 | In addition to the features described in V2.0, the robot can evade water that is offset to its left. The robot uses CSL to avoid water offset to the left of the robot by decreasing the Side Wall Target parameter of the base traversal algorithm and increasing the speed of the left wheel to steer the robot right around the robot.. |
| V2.2 | In addition to the features described up to and including V2.1, the robot can now use both sensors to avoid both the water on the left and the right side of the robot, utilizing the same mechanism described in V2.1 for both the CSR and CSL. Robot cannot effectively avoid water that is directly in front of it since this causes both the CSR and CSL water to detect water causing both wheel speeds to increase, meaning the robot does not turn. |

Table C.14: Motors and Sensor utilized in Base Traversal with Spanning Algorithm

| **Motor/Sensor** | **Description** |
|---|---|
| LML (Large Motor Left) | Large Motor mounted on the back left of the robot providing power to left wheel |
| LMR (Large Motor Right) | Large Motor mounted on the back right of the robot providing power to right wheel |
| USF (Ultrasonic Forward) | Polls forward facing wall until below left turn threshold |
| USS (Ultrasonic Side) | Polls side facing wall to maintain side distance target |
| CSL (Color Sensor Left) | Polls the ground on the left side of the robot to detect water |
| CSR (Color Sensor Right) | Polls cubes directly beneath to classify their type. |

Table C.15: Legend for Base Traversal and Spanning Visualization

| **Symbol** | **Meaning** |
|---|---|
| Black Arrow Large | Indicates robot's movement path and direction |
| Black Arrow Small | Direction of front of robot in span mode |
| Red Triangles | Scan FOV in Span Mode |

Table C.16: Detailed Iterations of Base Traversal with Spanning Algorithm

| **Iteration Number** | **Description** |
|---|---|
| V3.0 | The robot can return to the starting area after successfully completing one spiral, utilising an algorithm that parallels the original base traversal algorithm. The robot maintains a set distance from the side wall by polling the side wall with its USS. The robot polls the front wall using the USF and completes a turn after the wall is below the distance threshold. However in the updated algorithm, the distance threshold and side distance target are never updated and the robot's program is terminated after it reaches the fourth wall, finishing in the same corner it began in. The robot does not yet implement spanning. |
| V3.1 | In addition to the features described in V3.0 the robot now successfully implements spanning. After traveling a set distance, the robot pauses and begins to rotate to the left approximately 30°. After which point it rotates 30° to the right to return to the direction it was facing and begins travelling forward to complete the spiral described in V3.1. When integrated with the Block Detection algorithm, this spanning will allow the robot to scan a much larger portion of the available map. |

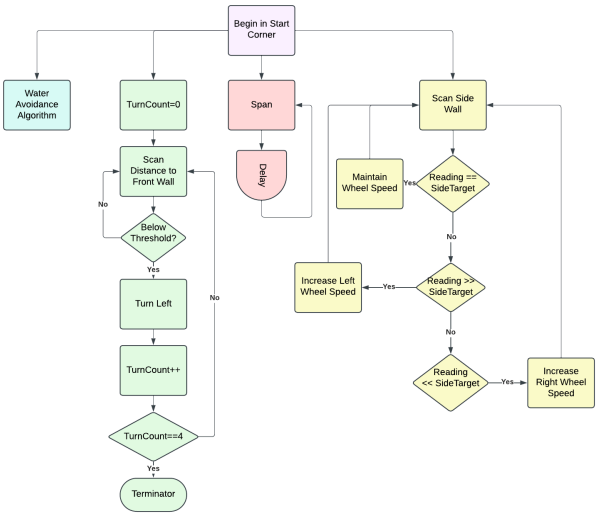



Figure C.5: Flowchart of Base Traversal with Spanning

Table C.17: Motors and Sensor utilized in Fast Return Home Algorithm

| **Motor/Sensor** | **Description** |
|---|---|
| LML (Large Motor Left) | Large Motor mounted on the back left of the robot providing power to left wheel |

| | |
|---|---|
| LMR (Large Motor Right) | Large Motor mounted on the back right of the robot providing power to right wheel |
| USF (Ultrasonic Forward) | Polls forward facing wall until below left turn threshold |

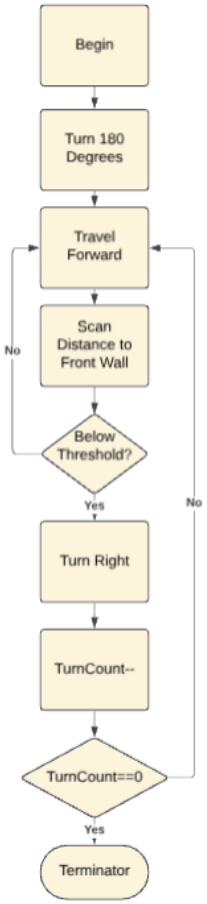



Figure C.6. Flowchart Visualization of Fast Return Home Algorithm

Table C.18: Detailed Iterations of Fast Return Home Algorithm

| **Iteration Number** | **Description** |
|---|---|
| V4.0 | The robot can successfully turn 180°. It then travels forward until the front ultrasonic sensor detects that the distance between the robot and wall is below the distance threshold. After the robot reaches the wall it turns 90° to the right after which point the program terminates. |
| V4.1 | The robot implements all features described in V4.0 but now keeps track of the number of left turns it has made up to this point while following the base traversal algorithm. The number of turns the robot has to make during the Fast Return Home Algorithm is equal to the number of turns made during the Base Traversal Algorithm so that the robot returns to the starting area. Before making a right turn during the Fast Return Home algorithm the robot checks to see if its number of remaining turns is 0. If it is, this indicates the robot has returned home and the program terminates. If it isn't the |

| | |
|---|---|
| | robot turns right and the number of remaining turns is decremented by one. It is noticed however that the robot often drifts to the right overtime causing it to often touch water on its return home. |
| V4.2 | In addition to the features described up to and including V4.1, the speed of the right wheel was increased so that on the robots return home it drives flush against the wall to its left, eliminating the possibility that it touches water during its return. |

Table C.19: Motors and Sensor utilized in Integration Algorithm

| **Motor/Sensor** | **Description** |
|---|---|
| LML (Large Motor Left) | Large Motor mounted on the back left of the robot providing power to left wheel |
| LMR (Large Motor Right) | Large Motor mounted on the back right of the robot providing power to right wheel |
| LMF (Large Motor Front) | Large Motor that rotates the USF after an object is detected to determine if detected object is cube or wall |
| USF (Ultrasonic Forward) | Polls forward facing wall until below left turn threshold |
| USS (Ultrasonic Side) | Polls side facing wall to maintain side distance target |
| CSL (Color Sensor Left) | Polls the ground on the left side of the robot to detect water |
| CSR (Color Sensor Right) | Polls cubes directly beneath to classify their type as well as polling the ground on the right side of the robot to detect water. |

Table C.20: Detailed Iterations of Integration Algorithm

| **Iteration Number** | **Description** |
|---|---|
| V1.0 | The robot begins in the corner of the map and follows the route specified in the Base Traversal Algorithm. Intermittently it spans from left to right. When the robot detects an object, the block detection algorithm is executed. If the object is determined to be the wall, the robot returns to following the Base Traversal Algorithm. If the object is determined to be a cube, the Block Approach and Block Classification algorithms are called. If the cube is classified as poop the Capture algorithm is executed and the robot returns to following the Base Traversal Path until it encounters another cube or returns to the start area.The Fast Return Home algorithm is not yet integrated. |
| V1.1 | In addition to the features described in V1.0, the Fast Return Home algorithm was implemented. If the Classification algorithm determines the cube to be an obstacle the robot carries out one of two responses. If the obstacle is on the primary path, the robot executes the Fast Return Home Algorithm and the program terminates. If the obstacle is not on the primary path, the robot reverses and returns to the Base Traversal path without touching the cube. The robot continues until it either reaches the start area or detects another cube |

D. Testing
Doc D.1: Navigation Subsystem Test Document

# Navigation Subsystem Test Documentation

---

Author: Katrina Panwar
Date: November 8th, 2024
Testers: Marrec, William

Sensors and Motors

- Sensors:
    - USF (UltraSonic Forward): Detects walls in the forward-facing direction.
    - USS (UltraSonic Side): Detects walls on the side, allowing for distance adjustments.
- Motors:
    - LML (Large Motor Left): Mounted on the back left of the robot.
    - LMR (Large Motor Right): Mounted on the back right of the robot.

Document Overview:

This document outlines the iterative testing process for the initial spiral navigation prototype, aimed at evaluating and refining a navigation algorithm that enables the robot to follow a spiraling path inward, covering most of the mapped area while using wall-based self-location.

The primary goal is to test an algorithm that uses multiple spiral iterations to progressively navigate inward, ensuring the robot maintains a specified distance from walls using forward (USF) and side (USS) ultrasonic sensors. The robot will rely on these sensors to detect obstacles, make 90-degree turns, and adjust its path accordingly, with specific thresholds for wall distance that increase incrementally as it progresses.

Doc D.1.1: Navigation Subsystem Test 1.0

# Navigation Subsystem Test 1.0

Hardware and Software Configuration:

- Navigation Script: base_traversal.py / V1.0
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Run the flute Python script base_traversal.py on the BrickPi.
2. Observe the robot as it follows the red line along the schematic map's border, using the USF sensor to detect front obstacles and the USS sensor to maintain a side distance.
3. The LML and LMR motors control straight-line movement and turns, ensuring balanced forward motion and precise 90-degree turns.
4. Record quantitative and qualitative data, assessing path adherence.

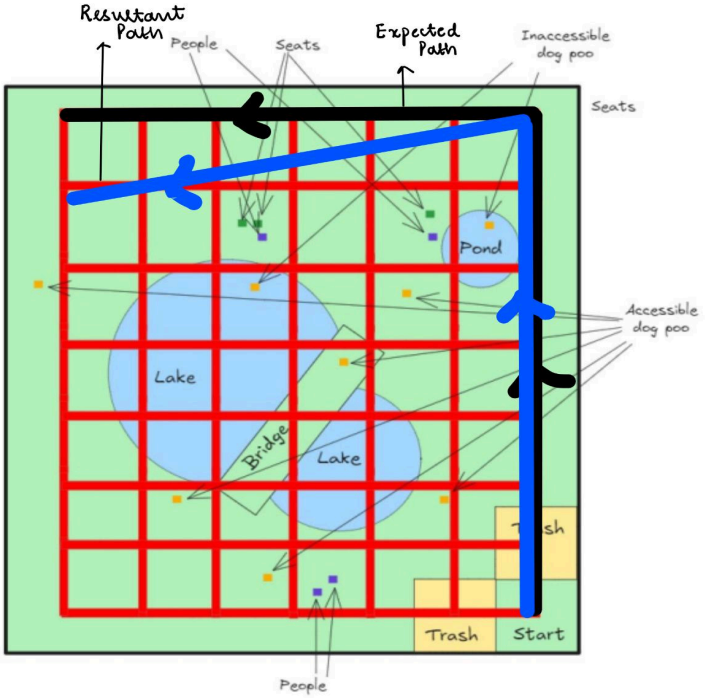



*Figure 1. Navigation Path*

Test Data:

The robot successfully followed the straight line and made a 90-degree turn at the corner as expected. However, instead of continuing along the intended path (black line, as shown in the picture), it deviated and followed an unintended path (blue line, as shown in the picture).

Doc D.1.2: Navigation Subsystem Test 1.1

# Navigation Subsystem Test 1.1

**Navigation Subsystem Test 1.1, November 8th, 2024**
**Purpose:** To verify the robot's ability to maintain a straight path after turns and follow the designated route without deviation.
**Procedure/Data:** see appendix Doc D.1.2: Navigation Subsystem Test 1.1
**Result:** The robot looped in place due to one motor stopping unexpectedly, highlighting synchronization issues in the software.
**Implications of the Test:** Turn control logic was refined to address motor synchronization issues, Emphasized the importance of precise motor coordination for straight-line movement, Identified the need for further testing under different speed and turn conditions.

Hardware and Software Configuration:

- Navigation Script: base_traversal.py / V1.1
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Run the flute Python script base_traversal.py on the BrickPi.
2. Observe the robot following the red line, with USF detecting front obstacles and USS maintaining alignment on the side.
3. Ensure the LML and LMR motors engage together after each turn to keep the robot moving straight.
4. Record quantitative and qualitative data, noting if the robot drifts off-course.

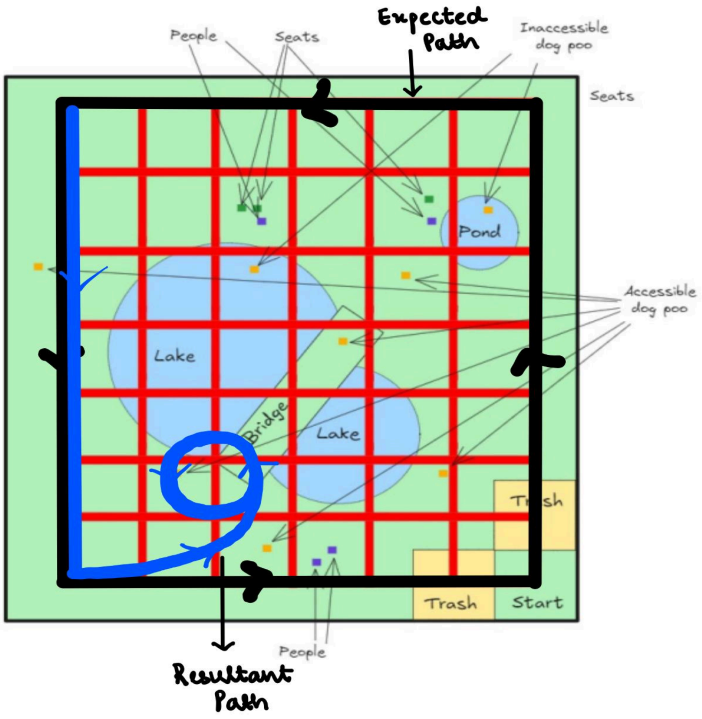



*Figure 2. Navigation Path*

Test Data:

The robot began moving in circles, with one wheel (LML) stopping while the other (LMR) continued to turn. This caused it to loop in place (shown by blue path on the map) rather than

proceeding forward, showing a need for adjustments in wheel synchronization and turn control logic.

Doc D.1.3: Navigation Subsystem Test 1.2

## Navigation Subsystem Test 1.2

Hardware and Software Configuration:

- Navigation Script: base_traversal.py / V1.2
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Run the flute Python script base_traversal.py on the BrickPi.
2. Observe the robot as it follows the red line, using USF for front obstacle detection and USS for side wall distance adjustments.
3. Monitor the LML and LMR motors to ensure both engage correctly after turns for smooth, forward motion.
4. Record quantitative and qualitative observations, verifying that the robot avoids looping in place and adheres to the path.

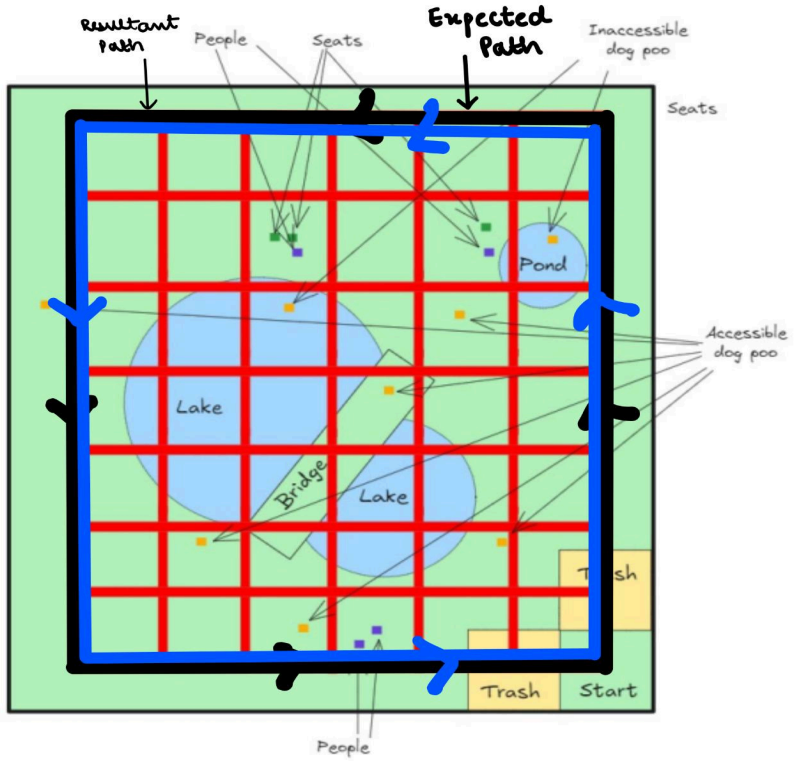



*Figure 3. Navigation Path*

Test Data:

When veering off course, the robot used USF and USS sensors to realign itself and stay on track. It successfully executed turns by stopping one wheel, adjusting the speed of the other, and then engaging both LML and LMR motors to proceed smoothly along the designated path.

Doc D.2: Navigation with Blocks Test Document

# Navigation with Blocks Test Document

---

Author: Karim Gaafar

Date: November 16th, 2024

Sensors and Motors

- Sensors:
    - USF (UltraSonic Forward): Detects walls in the forward-facing direction.
    - USS (UltraSonic Side): Detects walls on the side, allowing for distance adjustments.
- Motors:
    - LML (Large Motor Left): Mounted on the back left of the robot.
    - LMR (Large Motor Right): Mounted on the back right of the robot.

Software and Additional Components

Navigation Software

- Controls the robot's movement and path planning, ensuring efficient traversal and obstacle avoidance.

Cube Collection Cage

- A storage mechanism attached to the robot for collecting and holding cubes (dog poo) during traversal.

Testers: William, Marrec, Karim, Katrina

This document details the iterative testing process of the navigation subsystem while there are 6 cubes in the cage. The primary goal is to verify that the cubes do not impede the navigation ability of the robot.

Doc D.2.1: Navigation with Cube Test 1.0

# Navigation with Cube Test 1.0

Hardware and Software Configuration:

- Navigation Software / V1.0
- Navigation Hardware:
  - LML and LMR / V1.0
  - USS and USF / V1.0
  - Cube Collection Cage / V1.0

Test Procedure:

1. Place 6 cubes in the Collection Cage
2. Start the Navigation Software
3. Note after every turn the behavior of the robot
4. If a turn is not successfully completed, terminate the testing
5. Upon 4 turns, the test is considered successful

Test Data:

| Time | Behavior | Success |
|---|---|---|
| Turn 0 | Robot overcorrected by 90 degrees and got lost | No |

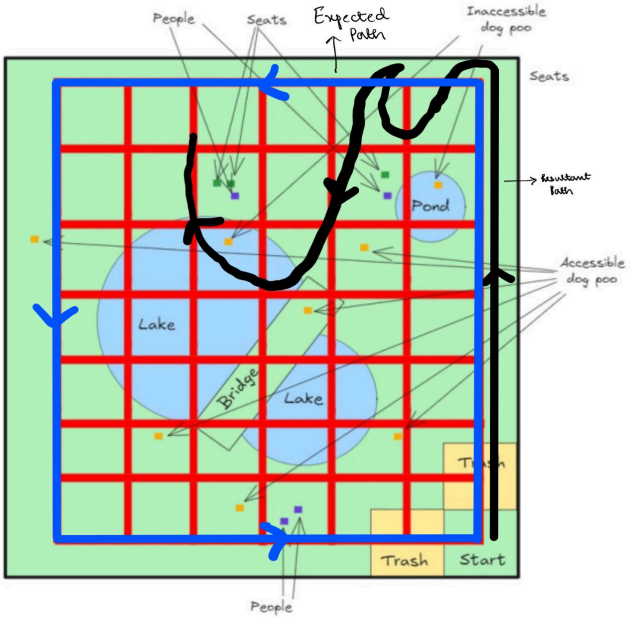



*Figure 1. Robot Path Map*

# Navigation with Cube Test 1.1

**Navigation with Cube Test 1.1, November 16th, 2024**

**Purpose:** To ensure smooth navigation with cubes and prevent prolonged path corrections.
**Procedure/Data:** see appendix Doc D.2.2: Navigation with Cube Test 1.1
**Result:** The robot started well but failed at the second turn, requiring significant corrections due to sensor misalignment.
**Implications of the Test:** Adjusted sensor thresholds for detecting distances more accurately, Identified the need for dynamic adjustments based on load weight, Refined the navigation algorithm to minimize overcorrection.

Hardware and Software Configuration:

- Navigation Software / V1.1
- Navigation Hardware:
    - LML and LMR / V1.0
    - USS and USF / V1.0
    - Cube Collection Cage / V1.0

Test Procedure:

1. Place 6 cubes in the Collection Cage
2. Start the Navigation Software
3. Note after every turn the behavior of the robot
4. If a turn is not successfully completed, terminate the testing
5. Upon 4 turns, the test is considered successful

Test Data:

| **Time** | **Behavior** | **Success** |
|---|---|---|
| Turn 0 | Follows the wall | Yes |
| Turn 1 | Spent too long correcting the path, never went straight | No |

Doc D.2.3: Navigation with Blocks Test 1.2

# Navigation with Cube Test 1.2

Hardware and Software Configuration:

- Navigation Software / V1.2
- Navigation Hardware:
    - LML and LMR / V1.0
    - USS and USF / V1.0
    - Cube Collection Cage / V1.0

Test Procedure:

1. Place 6 cubes in the Collection Cage
2. Start the Navigation Software
3. Note after every turn the behavior of the robot
4. If a turn is not successfully completed, terminate the testing
5. Upon 4 turns, the test is considered successful

Test Data:

| **Time** | **Behavior** | **Success** |
|---|---|---|
| Turn 0 | Follows the wall | Yes |
| Turn 1 | Robot turned more in place and quickly corrected | Yes |
| Turn 2 | Robot turned more in place and quickly corrected | Yes |
| Turn 3 | Robot turned more in place and quickly corrected | Yes |
| Turn 4 | Robot turned more in place and quickly corrected | Yes |

# Color Sensing Test Document

---

Author: Katrina Panwar

Date: November 18th, 2024

Sensors and Motors

- Sensors:
    - USF (UltraSonic Forward): Detects walls in the forward-facing direction.
    - USS (UltraSonic Side): Detects walls on the side, allowing for distance adjustments.
    - CSR ( Colour Sensor Right ): Identifies cubes (dog poo) and obstacles (e.g., people or chairs).
    - CSL (Color Sensor Left ): Detects surfaces (ground, grid lines, water, trash cans) for navigation.
- Motors:
    - LML (Large Motor Left): Mounted on the back left of the robot.
    - LMR (Large Motor Right): Mounted on the back right of the robot.

Testers: William, Marrec, Karim, Katrina

Document Overview:
This document outlines the iterative testing process for the color sensing subsystem, aimed at evaluating and refining the robot's ability to detect and classify various surfaces and objects accurately while in motion. The primary goal is to ensure reliable color classification by collecting raw RGB data, normalizing it, and comparing it to pre-defined normalized color centers using Euclidean distance calculations. The testing process will progressively address the classification of key environmental elements such as ground, grid lines, water, trash cans, and obstacles, as well as cubes representing dog poo. Initial iterations will focus on basic surface detection, while subsequent iterations will extend to object classification and edge case resolution, ensuring the subsystem is robust and performs reliably during full navigation tasks.

Doc D.3.1: Color Sensing Test 1.0

# Color Sensing Subsystem Test 1.0

Hardware and Software Configuration:

- Color Sensor Script:
    1. Collects RGB values from the color sensor during traversal - collect_color_sensor_data.py (V1.0).
    2. Processes collected RGB values, normalizes them, and classifies colors based on Euclidean distance to pre-defined normalized color centers - test_color_classification.py (V1.0).
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Position the robot on various test surfaces (e.g., grid lines, ground, colored blocks).
2. Run the collect_color_sensor_data.py and base_traversal.py scripts.
3. The robot collects raw RGB data from the color sensor during movement.
4. Normalize RGB values using normalize_color_sensor_data.
5. Classify colors using:
    a. Left Sensor (CSL): CSL_calculate_closest_color.
    b. Right Sensor (CSR): CSR_calculate_closest_color.
6. Calculate the Euclidean distance to pre-defined color centers and classify the closest match.
7. Output in real-time:
    a. Raw RGB values.
    b. Normalized RGB values.
    c. Closest detected color and distance.
8. The robot continuously collects and classifies color data, providing feedback until stopped.
9. Record data to assess color detection performance.

Test Data:

| **Actual Color** | **Detected Color** | **Success** |
|---|---|---|
| Ground (Green) | Ground | Yes |
| Grid (Red) | Grid | Yes |
| Water (Blue) | Water | Yes |
| Trash (Yellow) | Trash can area for disposal | Yes |
| Orange Block | Dog poo for collection. | Yes |
| Yellow Block | Dog poo for collection | Yes |
| Purple Block | Stationary people/obstacles | Yes |
| Green Block | Stationary seats/obstacles | Yes |

```
5
6  def CSL_calculate_closest_color(r, g, b):
7      # Use the normalized values
8      sample = (r, g, b)
9
10     # Ground Colors
11     ground_color_center = (0.338, 0.549, 0.113, "GROUND")  # Green
12     grid_color_center = (0.796, 0.112, 0.091, "GRID")  # Red
13     water_color_center = (0.202, 0.255, 0.543, "WATER")  # Blue
14     trash_color_center = (0.545, 0.379, 0.076, "TRASH")  # Yellow
15
16     color_center_array = [grid_color_center, water_color_center,
           trash_color_center, ground_color_center]
17
18     # distance, color name
19     closest_color = (10000, "NONE")
20
```

```
CSL: The closest color to (0.51, 0.33, 0.06) is TRASH with distance 0
    .06230569797378086

=== Code Execution Successful ===
```

*Figure 2. The expected output was ground but resultant was trash*

# Water Avoidance Traversal with Rotation Test

---

Author: Katrina Panwar, Karim Gaafar

Date: November 25th, 2024

Sensors and Motors

- Sensors:
    - USF (UltraSonic Forward): Detects walls in the forward-facing direction.
    - USS (UltraSonic Side): Detects walls on the side, allowing for distance adjustments.
    - CSR ( Colour Sensor Right ): Identifies cubes (dog poo) and obstacles (e.g., people or chairs).
    - CSL (Color Sensor Left ): Detects surfaces (ground, grid lines, water) for navigation.
- Motors:
    - LML (Large Motor Left): Mounted on the back left of the robot.
    - LMR (Large Motor Right): Mounted on the back right of the robot.

Testers: William, Marrec, Karim, Katrina

Document Overview: This document outlines a series of structured tests for the **Water Avoidance Traversal with Rotation** navigation system designed to evaluate and improve the robot's ability to navigate complex environments. The testing focuses on the robot's ability to detect, differentiate, and respond to environmental challenges such as water, obstacles, and cubes, ensuring robust and reliable navigation behavior.

The goal of the tests is to assess the robot's performance under various scenarios, identify shortcomings in the existing navigation algorithm, and propose iterative improvements for more effective traversal. Each test is structured to target a specific navigation scenario, with clear goals, hardware and software configurations, procedural steps, observed behaviors, and actionable next steps.

Doc D.4.1: Water Avoidance Traversal with Rotation Test 1.0

# Water Avoidance Traversal with Rotation Test 1.0

Hardware and Software Configuration:

- Water Avoidance Traversal with Rotation Script:
    1. water_avoidance_traversal_with_rotation.py/ V1.0
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Place the robot at the starting position.
2. Initialize the navigation software.
3. Observe the robot's behavior during navigation of the outer loop:
    a. Avoidance of water.
    b. Proper rotation when obstacles are detected.
    c. Continuation of the path post-rotation.
4. Terminate testing upon completion of the outer loop and return to the initial position.

Test Data:

| Turn | Behaviour | Success |
|---|---|---|
| Turn 0 | Avoids water and follows the wall | Yes |
| Turn 1 | Rotates Properly | Yes |
| Turn 2 | Avoids water and follows the path | Yes |
| Turn 3 | Returns to Initial Position | Yes |

Doc D.4.2: Water Avoidance Traversal with Rotation Test 1.1

# Water Avoidance Traversal with Rotation Test 1.1

Hardware and Software Configuration:

- Water Avoidance Traversal with Rotation Script:
    1. water_avoidance_traversal_with_rotation.py/ V1.1
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Place the cube at different locations on the map.
2. Start the navigation software.
3. Observe the robot as it encounters the cube and evaluates its surroundings.
4. Record whether the robot differentiates the cube correctly and avoids collision with the wall.

Test Data:

| Location | Behaviour | Success |
|---|---|---|
| Near the water | Avoids water and successfully detects cube | Yes |
| Right in front on ultrasonic sensor | Successfully detects the cube | Yes |
| Anywhere on the map other than edge | Successfully detects the cube | Yes |
| Near the edge (Close to the wall) | Detects the cube but doesn't stop and runs into the wall | No |

Doc D.4.3: Water Avoidance Traversal with Rotation Test 1.2

# Water Avoidance Traversal with Rotation Test 1.2

Hardware and Software Configuration:

- Water Avoidance Traversal with Rotation Script:
    1. water_avoidance_traversal_with_rotation.py/ V1.2
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Place the robot at the starting position of the outer loop.
2. Initiate the navigation software.
3. Observe when the robot completes the outer loop and begin to start the inner loop.
4. Observe the robot when it encounters water directly ahead (both sensors detecting water).
5. Note its behavior and determine if it avoids water or fails to adjust.

Test Data:

| Turn | Behaviour | Success |
|---|---|---|
| Turn 0 | Detects water head-on | No |

# Cube Locator and Differentiation Test

---

Author: Katrina Panwar, Karim Gaafar

Date: November 25th, 2024

Sensors and Motors

- Sensors:
    - USF (UltraSonic Forward): Detects walls in the forward-facing direction.
    - USS (UltraSonic Side): Detects walls on the side, allowing for distance adjustments.
    - CSR ( Colour Sensor Right ): Identifies cubes (dog poo) and obstacles (e.g., people or chairs).
    - CSL (Color Sensor Left ): Detects surfaces (ground, grid lines, water) for navigation.
- Motors:
    - LML (Large Motor Left): Mounted on the back left of the robot.
    - LMR (Large Motor Right): Mounted on the back right of the robot.

Testers: William, Marrec, Karim, Katrina

Document Overview: This document provides a detailed assessment of the **Cube Locator and Differentiation** functionality, focusing on the robot's ability to detect and classify objects using its ultrasonic and color sensors. The objective is to evaluate the system's performance in identifying cubes and differentiating between obstacles (e.g., chairs or people) and poop, ensuring the robot can navigate and respond appropriately in various scenarios.

The tests are designed to analyze the robot's behavior when cubes are placed in different positions relative to the wall, such as far from the wall and close to the wall. By simulating real-world scenarios, the tests aim to uncover limitations in detection, classification, and navigation algorithms and identify areas for improvement.

Doc D.5.1: Cube Locator and Differentiation Test 1.0

# Cube Locator and Differentiation Test 1.0

Hardware and Software Configuration:

- Cube Locator and Differentiation Script:
    1. cube_locator.py/ V1.0
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Place the robot at a predefined starting position.
2. Place cubes representing poop and obstacles at different locations (far from the wall and close to the wall).
3. Start the navigation software and let the robot detect the cube using the ultrasonic sensor.
4. Allow the color sensor to analyze the cube and classify it as poop or an obstacle.
5. Record whether the robot successfully identifies and differentiates between the two.

Test Data:

Test Table1. Far from wall

| Task | Success |
|---|---|
| Detects cube | Yes |
| Locates cube | Yes |
| Scans the cube | Yes |
| Does not hit the cube | Sometimes |

Test Table2. Close to wall

| Task | Success |
|---|---|
| Detects cube | Yes |
| Locates cube | No |
| Scans the cube | No |
| Does not hit the cube | No |

# Integration Test

---

Author: Katrina Panwar, Karim Gaafar

Date: November 28th, 2024

Sensors and Motors

- Sensors:
    - USF (UltraSonic Forward): Detects walls in the forward-facing direction.
    - USS (UltraSonic Side): Detects walls on the side, allowing for distance adjustments.
    - CSR ( Colour Sensor Right ): Identifies cubes (dog poo) and obstacles (e.g., people or chairs).
    - CSL (Color Sensor Left ): Detects surfaces (ground, grid lines, water) for navigation.
- Motors:
    - LML (Large Motor Left): Mounted on the back left of the robot.
    - LMR (Large Motor Right): Mounted on the back right of the robot.

Testers: William, Marrec, Karim, Katrina

Document Overview: This document provides a detailed assessment of the **Cube Locator and Differentiation** functionality, focusing on the robot's ability to detect and classify objects using its ultrasonic and color sensors. The objective is to evaluate the system's performance in identifying cubes and differentiating between obstacles (e.g., chairs or people) and poop, ensuring the robot can navigate and respond appropriately in various scenarios.

The tests are designed to analyze the robot's behavior when cubes are placed in different positions relative to the wall, such as far from the wall and close to the wall. By simulating real-world scenarios, the tests aim to uncover limitations in detection, classification, and navigation algorithms and identify areas for improvement.

Doc D.6.1: Integration Test 1.0

# Integration Test 1.0

Hardware and Software Configuration:

- Full Integration test Script:
    1. first_full_integration.py/ V1.0
- Controller Hardware: Robot / V1.0

Test Procedure:

**Phase 1: Start and Navigation**

1. Power on the robot and start the first_full_integration.py script.
2. Observe the robot's behavior as it moves:
    - Follows the grid path using navigation motors (LML and LMR).
    - Adjusts path based on sensor inputs to avoid obstacles.

**Phase 2: Water Avoidance**

1. Verify the robot detects water using ultrasonic sensors (USF, USS) and:
    - Avoids water by rerouting to the nearest alternative path.
    - Re-aligns with the main path after bypassing the water zone.

**Phase 3: Cube Detection**

1. Place cubes randomly on the grid (some near walls and others in open areas).
2. Confirm the robot detects cubes using color sensors (CSR, CSL).
3. Validate that it distinguishes cubes from walls or other obstacles:
    - Identifies cubes as either "poop" or "obstacle."

**Phase 4: Cube Pickup**

1. Ensure the robot initiates the pickup mechanism when a cube is detected.
2. Verify the robotic arm mechanism picks up the cube securely:
    - Check if the cube is placed in the collection cage or designated area on the robot.

**Phase 5: Cube Avoidance**

1. Introduce cubes near obstacles (e.g., walls, water zones).
2. Confirm the robot navigates around these cubes without collisions:
    - If the cube is not "poop," the robot avoids it.
    - If the cube is "poop," the robot picks it up.

**Phase 6: Return to Trash**

1. After collecting all cubes, observe the robot as it:
    - Navigates back to the trash can at the predefined position (e.g., grid (0,0)).
    - Avoids obstacles and water zones during the return trip.
2. Confirm the robot releases the collected cubes into the trash area.

Test Data:

| Phase | task | Success |
|---|---|---|
| Phase 1: Start and Navigation | Robot starts and moves forward on grid | Yes |
| | Robot maintains overall path alignment | Yes |
| Phase 2: Water Avoidance | Detects water using sensors | Yes |
| | Avoids water and re-aligns with path | Yes |
| Phase 3: Cube Detection | Detects cubes in open areas | Yes |
| | Detects cubes near walls | No |
| | Differentiates poop from obstacles | Yes |
| Phase 4: Cube Pickup | Picks up cubes from grid | Yes |
| | Secures cubes in collection cage | Yes |
| Phase 5: Cube Avoidance | Avoids non-poop cubes | Yes |
| | Avoids cubes near water zones | Yes |
| | If encounters cube, then turn around and move along the wall and cover the grid from the other side | No |
| Phase 6: Return to Trash | After collecting Cube on the outer grid from both sides, navigates back to trash area | No |

# Integration Test 1.1

Hardware and Software Configuration:

- Full Integration test Script:
    2. first_full_integration.py/ V1.0
- Controller Hardware: Robot / V1.0

Test Procedure:

1. Place poop cubes on the outer ring of the map.
2. Place a non-poop cube at the end of the outer ring.
3. Start the first_full_integration.py script.
4. Observe if the cube was detected
5. Observe if the cube was scanned.
6. Observe if the cube was not picked up.
7. Observe if the robot returns to trash.
8. Terminate the script.

Test Data:

| **Task** | **Success Percentage** |
|---|---|
| **All cubes were detected** | 50% |
| **All cubes were scanned** | 50% |
| **All cubes were picked up** | 66% |
| **Robot returns to trash** | Y |

## E. Performance and Capabilities

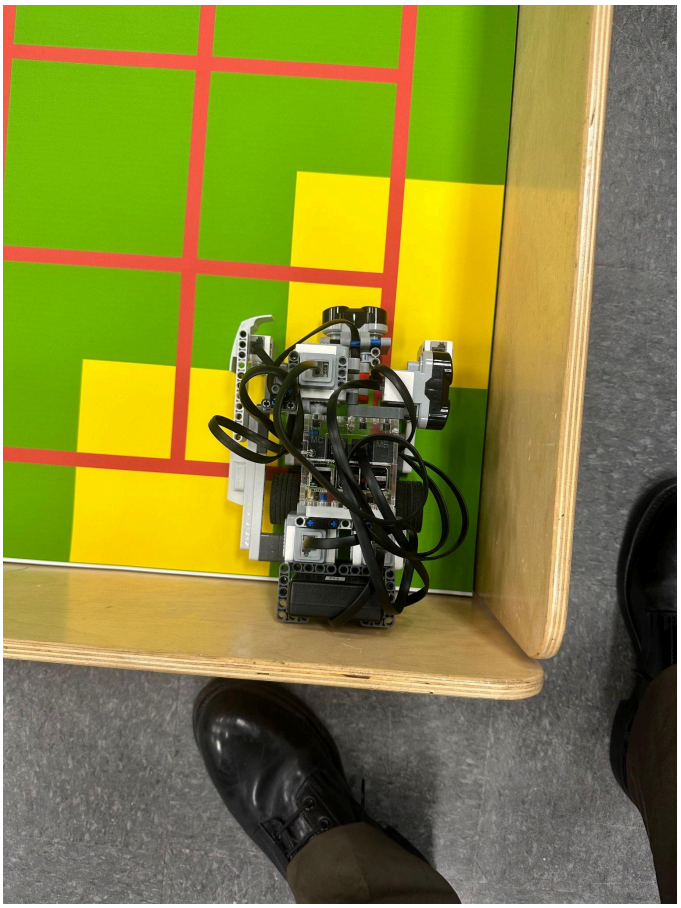


Image E.1. Robot Starting position

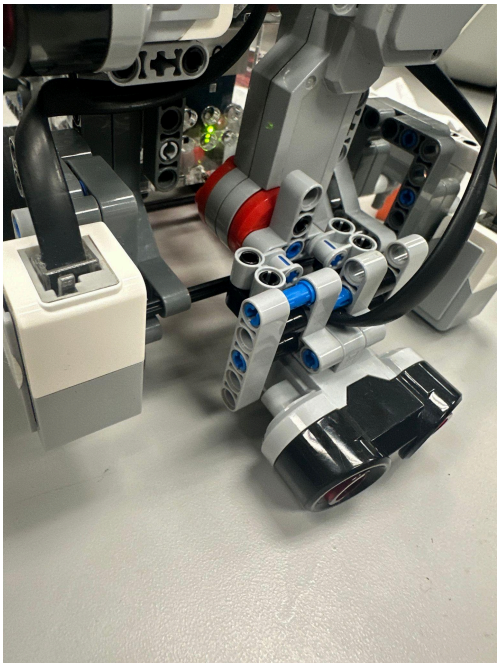


Image E.2. Starting Position of Front Ultrasonic Sensor