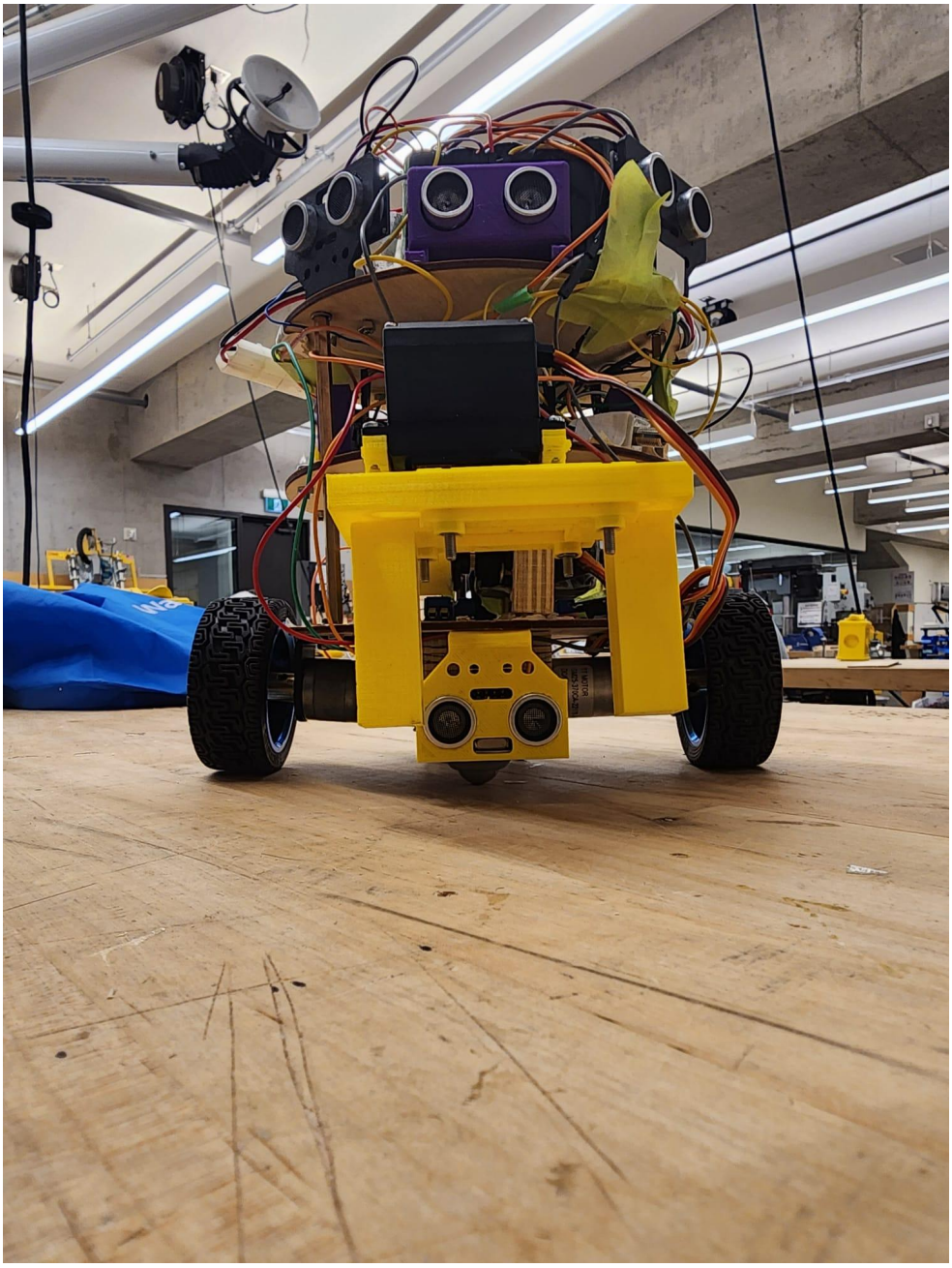
MIE444 Final Report



Yawar Ashraf 1006293672
Nishant Kumar 1005750849
Adi Bhattacharji 1005871629
Maanhar Singh 1005786197

Executive Summary

Our robotic navigation system employed a combination of ultrasonic sensors, Arduino programming, and a modular code structure to address challenges in obstacle avoidance, localization, and block delivery. The initial design utilized four ultrasonic sensors but faced issues with the robot's tendency to veer right. To counter this, additional sensors were added at 45-degree angles, significantly improving obstacle avoidance.

Localization initially posed challenges with MATLAB issues and particle filter complexities. A strategic shift to a semi hard-coded algorithm, integrating wall-following and rotation, proved effective. Our block delivery strategy utilized a gripper system with a rack and pinion mechanism, driven by low-cost servo motors. The robot's movement and block handling were governed by a meticulous code structure with obstacle avoidance, sensor alignment, and precise block detection.

During testing, the obstacle avoidance strategy demonstrated success in the first trial but faced challenges in Milestone 3, highlighting the importance of robust threshold values and comprehensive testing. The localization strategy faced initial hurdles but achieved success with a semi hard-coded approach. Block delivery encountered issues in block detection, particularly in misinterpreting maze corners, yet achieved success with autonomous navigation in Trial 3.

The modular code design facilitated efficient debugging and adaptability, overcoming unexpected challenges. Despite challenges, our robotic system demonstrated notable achievements in obstacle avoidance, localization, and block delivery, showcasing its potential for further refinement and application in diverse scenarios. Ongoing improvements in sensor calibration and code optimization are essential for enhancing overall performance and reliability

1. Detailed Rover Control Strategy

1.1 Obstacle Avoidance Strategy

Our approach involved equipping the robot with ultrasonic sensors and programming it with an Arduino to smartly navigate and avoid obstacles in its path. Our initial strategy was simple yet promising. We mounted four HS-S04 ultrasonic sensors on our robot, each pointing in the relative directions that were perpendicular from each other as seen in Figure X.

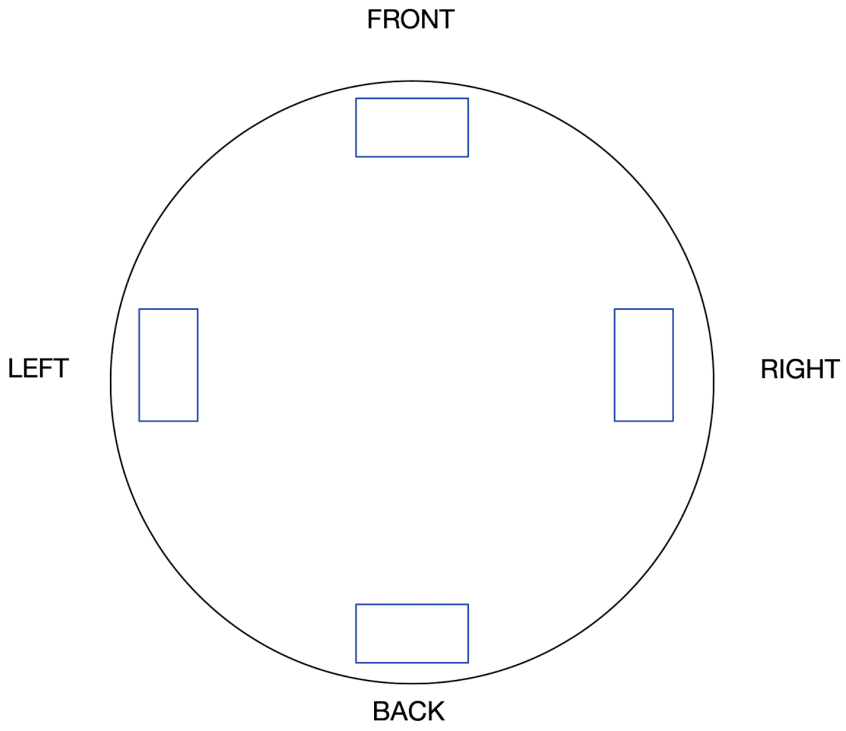


Figure 1: Initial Ultrasound Sensors on Robot

After installing these sensors, the avoidance strategy was fairly straightforward. The arduino was continuously getting real time readings from the sensors. It will continuously check for walls in all directions. It was coded so that the robot would move forward until its front-facing sensor detected an obstacle, specifically a wall, within 15 centimeters. If it sensed a wall directly ahead, it would check for walls to its left and right, again via ultrasonic sensor readings. Very similar to the front facing sensor, the threshold value for the left and right was 15 centimeters as well. Depending on where the walls were, it would turn 90 degrees in the opposite direction. If the robot was enclosed by walls on three sides, it would turn around completely and head back the way it came. A key part of our strategy was how the robot handled open spaces. If it

sensed a wall directly ahead and found no walls on either side, it would turn towards the side with more space. This is further explained in the flow chart in Figure X.

One of the most significant challenges we encountered in our project was ensuring that our robot moved in a straight line and one of our biggest mistakes was not fixing this until milestone 3. Encoders would have been a simple solution but we got to working on the problem too late to employ this solution. However, we did go through a thorough testing process to determine the root cause of the problem and came up with a fix. The fix wasn’t ideal but it worked.

Our first step was to carefully check all the parts of the robot. We wanted to understand why it was not moving straight and kept drifting to the right. After examining everything and swapping out various components, we thought that the issue might be due to the way the robot's weight was distributed.

To test this idea, we changed the robot's programming to see if altering the way the wheels moved would help. We tried making the right wheel move slower, and the left one faster for the same amount of time, but this did not solve the issue.Finally, we programmed the robot to move its left wheel for a set amount of time, then stop it. After that, we did the same thing with the right wheel. This approach allowed the robot to move forward in a straight line and stay facing the front.

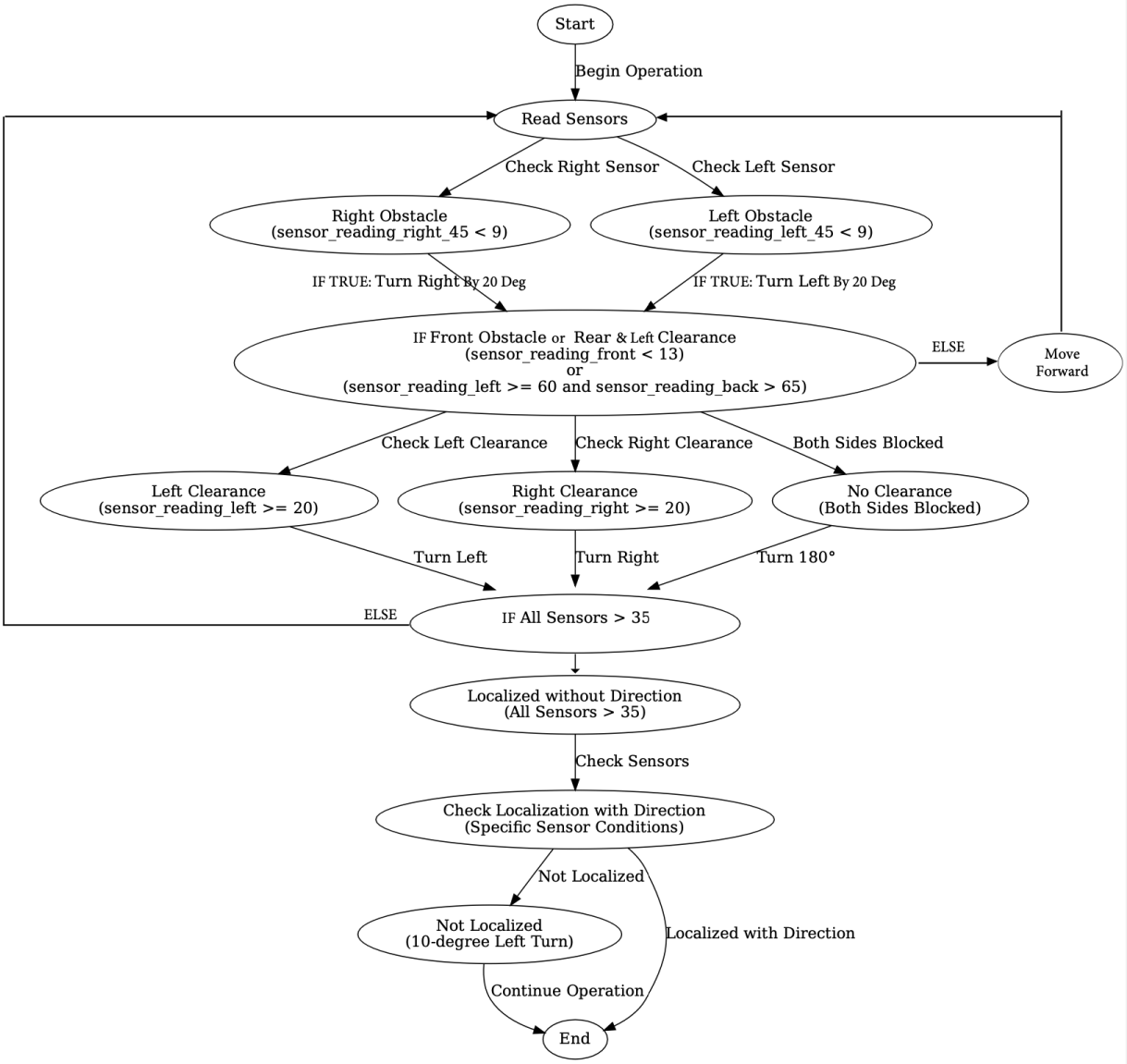


Figure 2: Flowchart of Robot Avoidance Strategy

Despite our planning, we encountered a major challenge: our robot had a strong tendency to veer right. It would mean that the rover continued to collide with the wall at a 45 degree angle. This meant that our sensor setup did not account for obstacles at 45 degree angles. To solve this, we added two more ultrasonic sensors, this time angled at 45 degrees to the front of the robot, facing front left and front right, seen in Figure X, which is the updated version of the sensors on the rover.

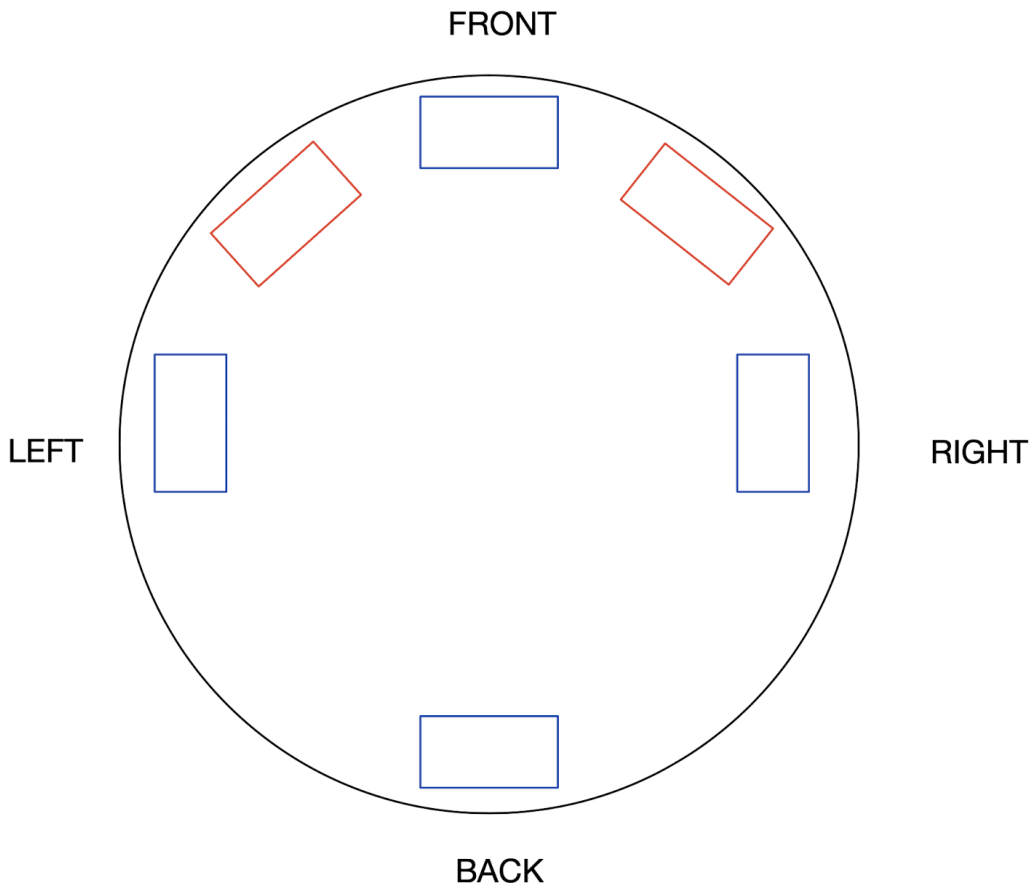


Figure 3: Updated Ultrasound Sensors on Robot

If the front left and front right sensor detected an obstacle within 6 centimeters, the robot would adjust its course slightly in the opposite direction. This adjustment was crucial and significantly improved our robot's obstacle avoidance strategy. For instance, if the northeast sensor detected a wall or an obstacle within 6 centimeters, the robot was programmed to correct its course by slightly turning to the left. This subtle adjustment played a key role in reducing the robot's tendency to drift rightward. Some of the unexpected improvements that it also provided was navigating tight spaces and corners more smoothly. Furthermore, the angled sensors were effective in identifying potential collisions with walls on the robot's side that were not recognized by the front-facing and side-facing sensors. Overall, improving the sensor coverage by adding the front left and front right sensors enabled the robot to maintain a straighter trajectory and avoid collisions, ensuring more reliable and accurate navigation through the maze.

Using this type of obstacle avoidance caused two different challenges. The first was one the most time-consuming aspects of our project, which was to fine-tune the threshold values for the sensors. This process was tedious and required extensive trial and error.

We ran the robot from various starting points in the maze, observed its performance, and noted where and why it crashed. Each crash led to adjustments in our threshold values, ensuring the robot wouldn't fail in the same spot again. This was crucial as different points in the maze affected the sensor readings differently, making it a time consuming and nuanced task.

The other challenge we faced was specific yet infrequent: the robot occasionally struggled with corners, especially when approaching one at a 22.5-degree angle. This angle was out of the detection ranges of our front and angled sensors. Since this was a rare occurrence, it was still a potential issue. However, considering the low probability of this happening, we accepted this risk as part of our design.

1.2 Localization and Navigation Strategy

When formulating our localization and navigation strategy, we encountered a noteworthy challenge when implementing the histogram localization code. There were issues running MATLAB in a local environment, which impeded the practicality of this approach. This forced an entire revision of our methodology. Furthermore, our original approach involved subjecting the robot to multiple simulations with the anticipation that one would converge to the correct value, offering a reliable estimation of the robot's orientation. However, the inherent risk in this method lay in exposing the rover to subtle biases in the real-world environment, subsequently compromising the reliability of localization readings derived from the histogram localization simulation.

In response to the multifaceted challenges, we made a strategic pivot towards deploying a particle filter, recognizing its inherent invariance to the robot's heading. This shift, while promising, was not without its intricacies. We grappled with the initial complexities of setting up a custom Simmer environment and developing the algorithm. Nevertheless, we maintained our belief in the advantages offered by the particle filter, seeing them as outweighing the challenges. However, the completion of the algorithm brought unexpected performance issues to light, laying fundamental flaws in our initial assumptions.

A crucial limitation surfaced during our analysis: the absence of a direction assignment for each sampled point. Our process of updating the confidence of each square during time steps and normalizing the probabilities for each square at the end of each time step led to a peculiar scenario. The robot appeared stuck due to a sudden jump in square probability to 100% after normalization, despite a reduction in the same time

step. Complicating matters further, issues resembling exploding weights and vanishing weights emerged, casting shadows on the reliability of the particle filter.

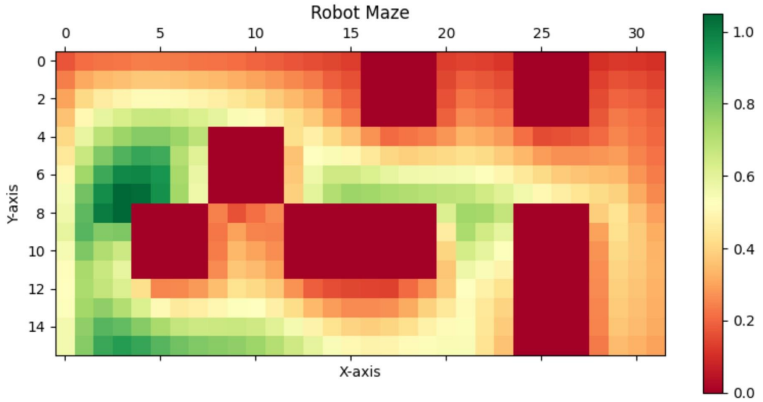


Figure 4: First iteration of the particle filter with exploding/vanishing weights

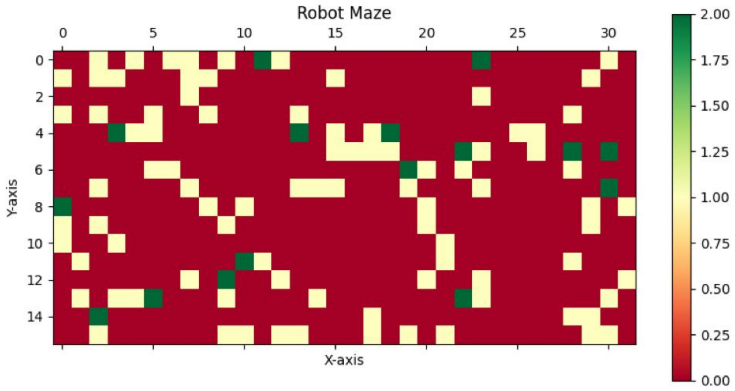


Figure 5: Fixing Exploding weights and going off of only the number of particles in a region. Experiencing poor convergence

Faced with these intricate problems and constrained by time (having less than 24 hours to devise a solution), we strategically decided to veer away from both the histogram localization and particle filter algorithms. Instead, we opted for a semi hard-coded algorithm, which centered on continuous robot movement until it reached a unique spot in the maze.

This selected algorithm seamlessly integrated a wall-following policy as the robot maneuvered through the maze, with a particular emphasis on following the right wall. With eight sensors in total – front, right, back, and left, complemented by a second set rotated at 45 degrees – the robot sought to detect itself in a localization square where

no walls were present on all four sides. Upon successfully localizing, the robot initiated a clockwise rotation to align itself northward, referencing the simmer environment.

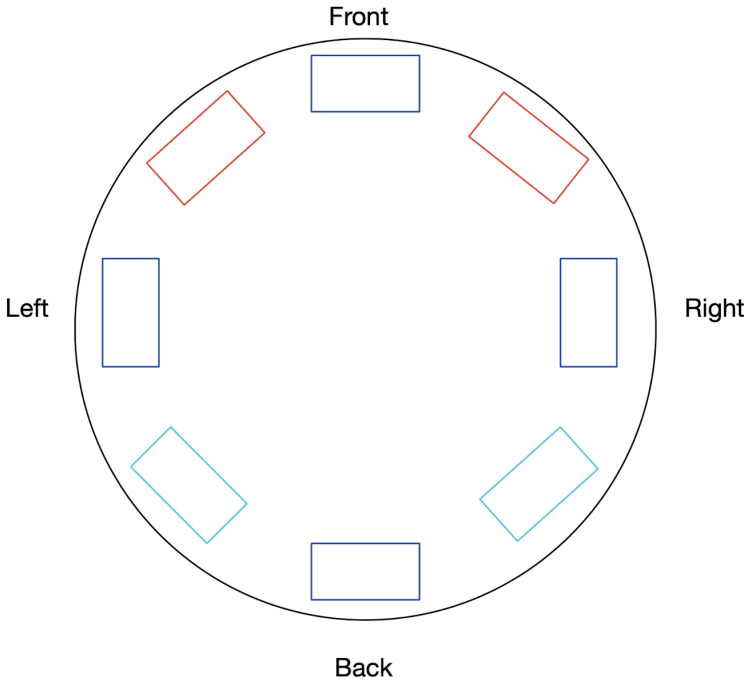


Figure 6. Added two more sensors to the back to try to detect our localization square.

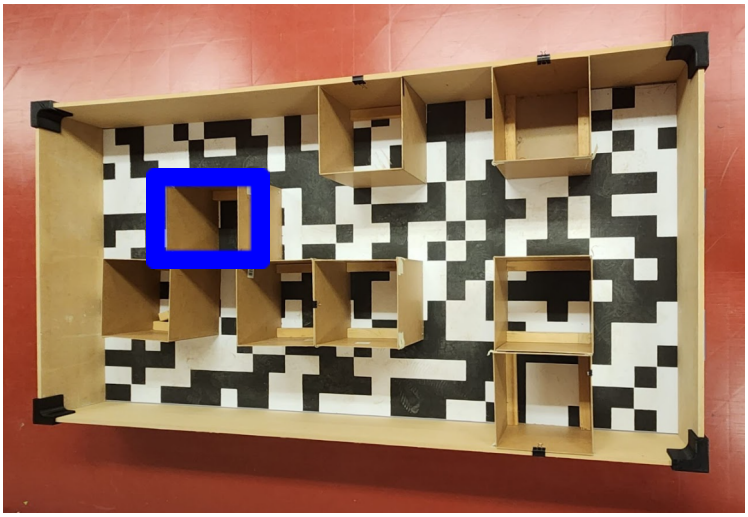


Figure 7: Localization Zone in the Maze labelled by blue square

The determination of cardinal direction thresholds, meticulously informed by extensive maze testing, played a pivotal role in this process. Successfully localized within the finite maze environment, the robot embarked on movements toward the loading zone based on sensor readings. This triggered middleware code to validate the robot's arrival in the loading zone, subsequently activating middleware code related to block detection and

delivery upon reaching this crucial point. This strategic shift towards a semi hard-coded algorithm allowed us to navigate the intricate challenges efficiently and make substantial progress within the time constraints, thereby ensuring the ultimate success of our overall robotic navigation system.

1.3 Block Delivery Strategy

The gripper system for the robot utilizes a rack and pinion system to provide the gripper's jaw movement mechanism. It consists of two parallel jaws driven by individual rack and pinion systems, enabling parallel and consistent movement while gripping the block. The parallel rack and pinion system is compact, making it a suitable design due to the limited space available for the gripper system on the rover.

The gripper claw utilizes a low-cost MG996R servo motor to drive the pinion gears to open and close the gripper claw. The servo-controlled aspect of the system ensures precise and programmable control over the jaw movement.

The robotic arm was controlled by another MG996R servo motor that was used to lift the gripper claw upwards and downwards to lift the block. The entire robotic gripper mechanism was 3D printed by using PLA. The use of a low-cost MG996R servo motor allows the rover to operate within the budget constraints without compromising performance.

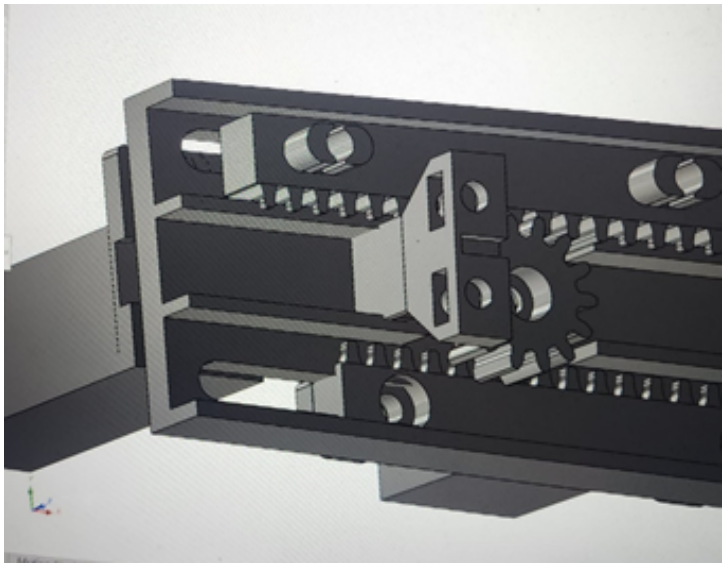


Figure 8: 3D CAD design of Gripper Claw, showing the parallel rack and pinion system

The strategy for handling block delivery involves entering the loading zone, grabbing and taking the block to the final drop-off point to eventually release the block. The code employs a combination of obstacle avoidance, sensor alignment through rotation, and block detection to achieve its goal. The primary control structure is a `while` loop that continues execution until the variable `blockpickup` becomes True. This variable serves as a flag indicating whether the robot has successfully picked up the block. The loop encapsulates the entire navigation and pickup logic, ensuring continuous execution until the desired objective is met.

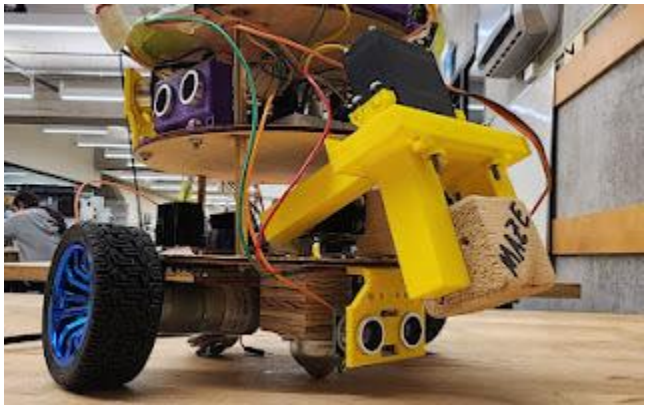


Figure 9: Picking up the block using the Rover

To grab the block, the rover would recognize that it's entering the loading zone. It then turns 90 degrees to orient itself from its center. To make sure that it sees the block, it turns by small rotational increments of 15 degrees until it realizes the direction where a block exists. There is a brief pause that allows the robot to adjust its orientation. To determine the direction of the block, the microcontroller subtracts the readings of the bottom and front sensors. It then compares the difference in the values to a specific threshold. If the difference falls below a specified threshold, the robot acknowledges the difference and pauses to read sensors another two times to make sure that the block has indeed been detected and the difference in sensor readings is not due to just noise. While rotating, a counter is employed that helps in tracking the number of times a significant difference between the bottom and top front sensor is detected. This is so that once it passes 360 degrees, it will move forward by half a step and redo the rotations by 15 degrees.

Upon detection, the rover would stop rotating and it would position itself so that the gripper points towards the middle of the block. When the difference between the top and bottom sensor has the same value for more than a specified iteration (our case was 3

sensor readings). The code exits out of the rotation loop and the rover is able to identify the direction of the block successfully.

Upon moving forward, the top front and bottom sensors are continuously providing distance values and the difference in their values is compared to a threshold. When the threshold value is less than 6.5 cm, the robot identifies that it has successfully reached the block. Now, the gripper will grab the block by opening the gripper claw, moving the arm down, close the gripper and move the arm up in order to lift the block.

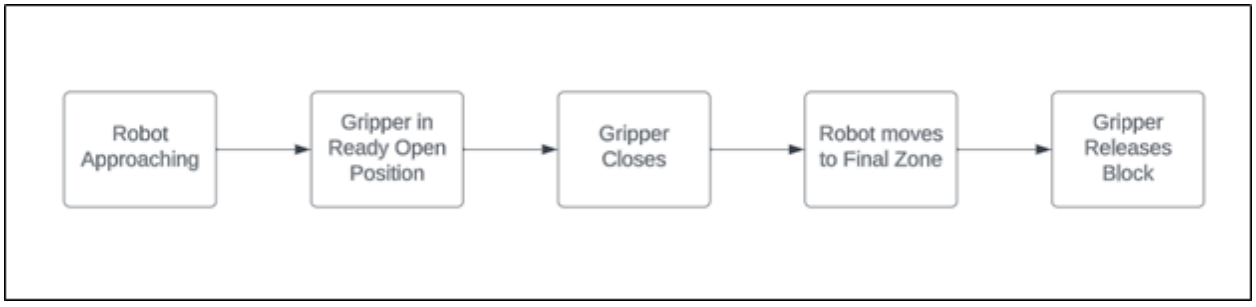


Figure 10: Flowchart of Gripper Strategy

By conducting thorough testing, we discovered that ultrasonic sensors exhibit subpar performance when encountering walls or blocks at an angle relative to the sensor. This was detrimental in detecting the block as we compared the difference in the measurement between the top and bottom sensor that would be compared to a threshold value.

As a result, we took into account the extreme cases when the block is at relative angles to the sensor and included a separate test case in the code. The rover would move to the side and then forwards so that the sensor is perpendicular to the block. The rover makes use of the ultrasonic sensor to realign itself with the block in the maze. Also, making sure that the block was initially placed in the maze parallel to the maze walls, helped in mitigating the issue.

In order to mitigate other issues and increase the overall speed of the robot, the group determined that utilizing the rover's high degree of rotational precision would be a quicker approach. This was taken advantage of by recording the total number of rotations the rover made. As it kept count, this would eventually be helped when the robot needed to head to the localization zone. It would then go to the drop-off zone based on code from said localization spot. Moreover, we included comments such as “we have the right direction” and “we have reached the block” to make programming and defining functions easier.

Once the block was secured, the rover would localize itself based on the current position and navigate to the user-defined drop zone. Upon confirming its location within the drop zone, the rover moved the arm down and opened the grabber to drop-off the block.

1.4 Integration

Our comprehensive code pipeline was meticulously structured in a modular fashion, consisting of distinct components at lower, middle, and higher levels. At the foundational lower level, we intricately implemented code to govern elemental robot movements such as forward, backward, rotation, sensor readings, motor stoppage, and the manipulation of the robot claw and robotic arm. The middle-level code was instrumental in orchestrating complex actions, including block pickup and drop-off, thorough sensor readings, and error correction movements. Simultaneously, the high-level code encapsulated sophisticated routines for detection, obstacle avoidance, localization, and block pickup and drop-off.

Our strategic approach to code design prioritized modularity, with a foresight into potential issues that could potentially disrupt the code's seamless functionality. Despite the discrete routines, we ensured a sequential implementation, strategically deploying flags to activate and deactivate specific sections based on the progression of the sequence. This modular structure not only facilitated ease of debugging but also adhered to an organized and systematic development approach.

In particular, our localization code was thoughtfully designed to minimize dependence on the robot's precise position throughout its operation. Initially, the robot embarked on a movement until it autonomously localized itself, as detailed earlier. Once successfully localized, the navigation was precisely directed towards the loading zone. This unique capability stemmed from the robot's knowledge of both its position and heading. Subsequently, the robot continued its movement towards the loading zone, with a specific flag activated to detect its arrival. Upon reaching the loading zone, control seamlessly transitioned to the block detection code, operating in a manner analogous to other major robotic functions.

The block detection code persisted in its endeavors to locate the block, maintaining the block pickup flag at a low state. The successful pickup of the block triggered a significant shift in the block pickup flag, setting it to a high state. This strategic decision promptly returned control to the localization code, allowing the robot to re-localize itself. Following this, the robot deftly navigated to the block drop-off location, with the

predetermined drop-off location and its corresponding flag manually activated before each test based on insights from the teaching team.

Post-localization, control seamlessly transitioned to the drop-off zone navigation code, initiating a meticulous three-step process for the block drop-off: the controlled lowering of the robotic arm, the precise opening of the claw, and the methodical raising of the robotic arm. Throughout this entire process, the obstacle avoidance code remained actively engaged, ensuring continuous functionality. However, our team encountered additional challenges related to ensuring seamless communication between different robot components. The tuning of serial communication delays became a critical consideration, striking a delicate balance between a high success rate of data packet delivery and avoiding unnecessary delays that could slow down the robot.

In light of these challenges, the team devoted additional effort to establish a proper control flow within the code base, necessitating a comprehensive refactoring of the code base and the implementation of a sophisticated flag system. This strategic decision aimed not only to identify specific sections of the robot exhibiting unexpected behavior but also to streamline control flow for more efficient problem-solving as a collaborative team effort. The modular design further proved instrumental in detecting and effectively resolving various edge cases that surfaced during rigorous testing runs, underscoring the robustness and adaptability of our code architecture.

2. Final Results

2.1 Obstacle Avoidance Strategy

In assessing the performance of our rover, particularly in terms of its obstacle avoidance capabilities, we observed a marked difference between its initial trial and the subsequent Milestone 3 challenge. The effectiveness of the obstacle avoidance strategy was notably influenced by factors such as code robustness and hardware readiness.

During the first trial, the rover's performance in obstacle avoidance was exemplary. It successfully navigated through 20 blocks in the maze, achieving a perfect score without a single collision. This success can be attributed to the meticulously fine-tuned threshold values of the sensors and the effective integration of additional sensors at strategic angles. The rover demonstrated an exceptional ability to adjust its course in

real-time, effectively avoiding walls and obstacles, and maintaining a straight path throughout the maze.

Contrastingly, during Milestone 3, the obstacle avoidance system did not perform as efficiently. The primary issue here was the lack of comprehensive testing, especially concerning the partially hardcoded segment of the navigation from the localization point to the loading zone. The threshold values in this section of the code were not as robust as those in the main obstacle avoidance algorithm, leading to minor collisions. This highlighted the importance of thorough testing across all code segments to ensure consistent performance under varying conditions.

Another critical factor that impacted the rover's performance during Milestone 3 was the insufficient battery charge. This resulted in the motors receiving less power than usual, causing the rover to move and turn inadequately. Such a deviation from the expected behavior significantly disrupted the calibration of the obstacle avoidance code. It underscored the necessity of ensuring hardware readiness, particularly in terms of power supply, to maintain the integrity of the programmed navigation and obstacle avoidance strategies.

2.2 Localization and Navigation Strategy

In Milestone 2, our robot was able to successfully localize itself within the Simmer environment. This milestone in our development process showcased the ability of the code to understand and navigate a simulated space with the robot. A key issue we faced was that the serial communication between the code and the actual robot had not been set up and we were unable to get the robot moving in the actual maze. Building on this, Milestone 3 brought about further advancements in localization capabilities. Leveraging the implementation of serial communication, the robot was able to operate within the physical maze environment. Throughout the first and second trials of Milestone 3, the robot autonomously localized itself within the maze, exhibiting a reliable understanding of its position and orientation. A key addition to the localization strategy was to ensure that the team tested multiple times in the actual maze and therefore we were able to identify many edge cases that allowed the robot to navigate reliably. Throughout the project one of our key goals was to establish bluetooth communication between the robot and the code however due to code environment issues we were unable to do so.

2.3 Block Delivery Strategy

In Milestone 3, the robot had the task of autonomously arriving at the loading zone with confirmation, picking up the block, and delivering it within a 7-minute timeframe. During the practice attempts, the robot faced challenges in accurately detecting the block. Despite navigating to the loading zone, it struggled to correctly identify the block.

Based on the ultrasonic sensor readings, the robot would misinterpret the corners of the maze as the block. For example, the robot mistakenly identified a corner as the block, executed the grabbing code, and proceeded to the drop-off point as if it had the load. This issue seemed to stem from a discrepancy between the gripper ultrasonic sensor and the front ultrasonic sensor when facing a wall. Also, at relative angles to the block, the sensor would display extremely high distance readings (12m), which made it difficult to detect the block.

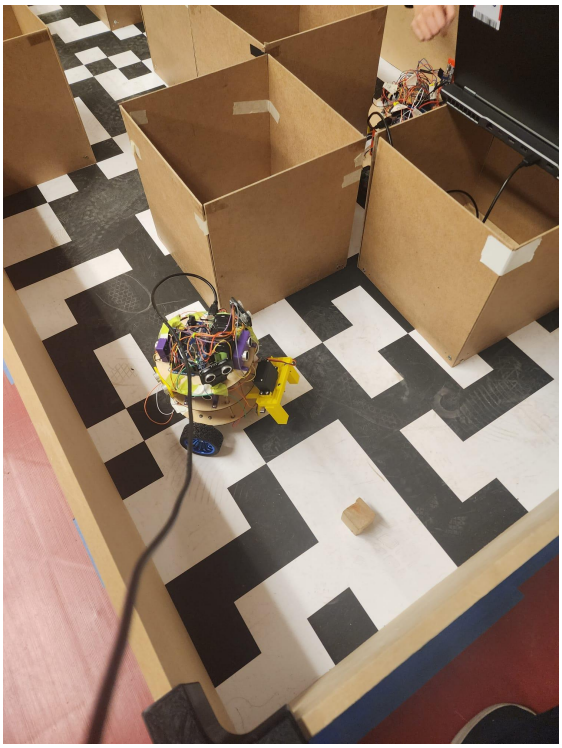


Figure 10: Detecting the block to pick it up. Notice the bias it has

In Trial 3, the robot was able to successfully identify, pick up, and deliver the block to the final drop-off point. Since the robot was controlled completely autonomously, it accomplished the task with the help of some slight adjustments to its orientation. Furthermore, the attachment of the gripper had fallen off when it got into the finish load, rather than just dropping the block.

2.4 Integration

In Milestone 3, our robot reached a noteworthy achievement by successfully navigating the entire process of picking up, identifying, and delivering a block, all while adeptly utilizing our obstacle avoidance code. This showcased the integration of various functionalities within our robotic system. However, the journey wasn't without its hurdles. We encountered challenges in the smooth transition of control between different parts of our code, resulting in delays during localization, block pickup, and subsequent localization for drop-off. The pace of our robot's movements proved notably sluggish, taking a considerable four minutes to reach the loading zone, leaving only a brief one-minute window for localization and transportation to the final drop-off zone.

To enhance our system's efficiency, we identified potential improvements to reduce these delays, aiming to elevate the overall responsiveness of our code. A specific challenge arose during the block pickup phase, exposing issues in the integration between straight-line driving and block detection. The robot displayed a bias towards the right, affecting its ability to align precisely with the block for successful pickup. This highlighted the need for further refinement in coordinating motor movements and sensor readings to ensure accurate alignment during critical tasks. Addressing these challenges in control flow and responsiveness, coupled with fine-tuning the integration between driving and block detection, is a pivotal focus as we strive to optimize the efficiency and overall performance of our robotic navigation system in subsequent iterations.

3. Discussion

Best Features:

In terms of notable characteristics, the implemented code pipeline stood out for its remarkable modularity, allowing for comprehensive control over the various aspects of the robot. The robot's equilibrium was another positive attribute, contributing to its overall stability during operations. The inclusion of castors played a crucial role in enhancing the robot's mobility and balance. Additionally, the design exhibited a simplicity in hard coding, facilitating ease of debugging—a feature that proved beneficial during the development process. The small and compact design, while advantageous in reducing the likelihood of collisions with walls, was also instrumental in navigating through confined spaces.

Worst Features:

The robot encountered difficulty maintaining a consistent straight trajectory primarily due to a discernible bias in the right motor, and the absence of encoders contributed to imprecise control. The issue of unreliable Bluetooth connectivity drew attention to the necessity of establishing robust communication protocols adaptable to diverse hardware configurations. On the mechanical front, concerns centered around the inadequately secured gripper and unresolved servo motor connection problems. Frequent collisions with walls highlighted the intricacies of collision avoidance mechanisms, and the limited range for block detection at specific angles emphasized the importance of refining the robot's sensing capabilities. The aesthetic aspect of the robot's design was not overlooked, with observations noting components haphazardly taped up and H-drives protruding, impacting the overall visual appeal of the robot. To effectively address these challenges and enhance the robot's overall performance, future iterations should prioritize comprehensive improvements encompassing hardware enhancements, software optimizations, and a refined system design approach.

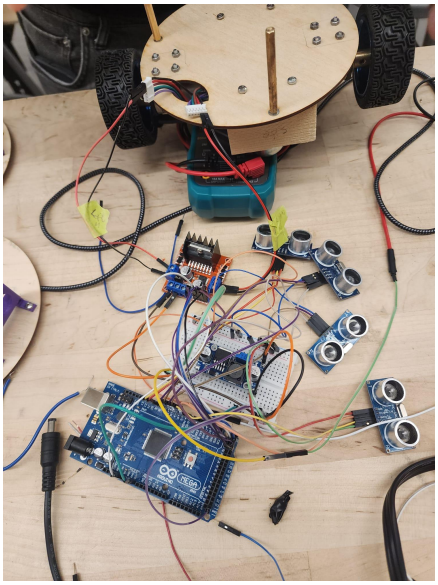


Figure 11: Testing parts to make sure everything works in tandem

Practical Solutions for Improvement:

In terms of practical solutions for enhancement, employing time-of-flight sensors to obtain reliable readings proved advantageous compared to using ultrasonic sensors, which presented challenges when detecting objects at angles. Considerations for the robot's platform design, particularly accounting for gripper attachment, could be incorporated. This involves reverse engineering to ensure compatibility between gripper and rover designs. Ensuring adequate space for complete mobility and the proper addition of components, such as the buck converter and h-bridge, became integral for seamless functionality.

Design Compared to Initial RFP and others:

The evolution of the design, informed by enhanced knowledge of power and electrical systems, showcased notable improvements. Complexities in the initially proposed gripper design led to a streamlined and more achievable version. Drawing inspiration from other groups, the incorporation of IR sensors at 45-degree angles offered a potential solution to minimize collisions and enhance block detection.

Future Project Considerations

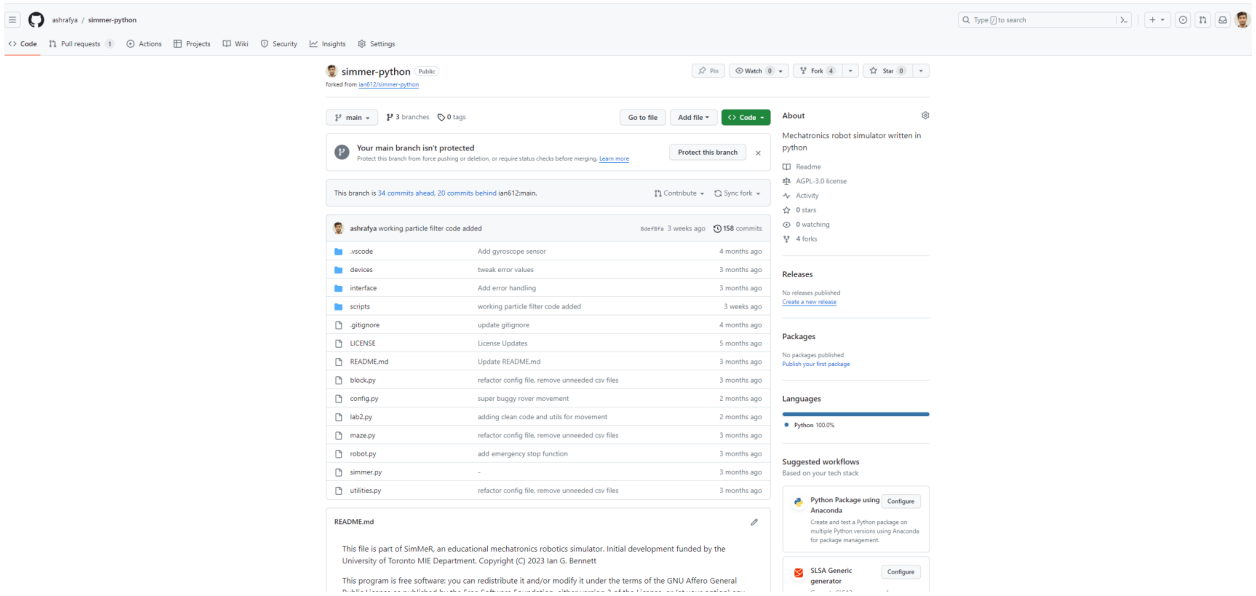
Reflecting on the project, several insights emerged for future endeavors. Addressing visible challenges promptly, such as implementing solutions for straight-line movement, would be a priority. Transitioning from Arduino to Python coding at an early stage and ensuring a stable power circuit through soldering were identified as strategic moves. Design considerations, like creating larger platforms for wiring maneuverability, were deemed beneficial. Ensuring the gripper's sufficient mounting and early integration into the design process were highlighted for improved functionality. Understanding the power circuit's limitations and potential failures underscored the need for thorough planning.

Project and Team Management:

In terms of project and team management, initiating regular team meetings and check-ins from the project's onset was deemed essential. Recognizing the consequences of delayed initiation on project timelines emphasized the need for timely planning and execution. Holding team members accountable for tasks and promoting fair task distribution were recognized as vital aspects of effective collaboration. The importance of team members actively contributing from the project's initiation highlighted challenges associated with mid-project contributions. Setting measurable goals, having open communication channels, and establishing expectations for timely updates in case of missed deadlines were identified as foundational for a successful team dynamic. The team's core values centered on sincerity in work ethic, understanding the importance of investing ample time when needed, and fostering open communication to avoid underestimation of individual contributions.

In summary, the project exhibited a blend of commendable features and areas for improvement, providing valuable insights into both technical and team management aspects. The journey encompassed challenges, iterative improvements, and a wealth of experiential learning that would undeniably shape future endeavors in a similar domain.

Appendix A: Code



The Code can be found here: https://github.com/ashrafya/simmer-python