Mechatronics Design Report

Phase C

Group 21 - OnTechUwU

METE 4100 | Meaghan Charest-Finn

#	Last Name	First Name	Student Number	Signature
1	Elvin	Bryce	100696601	Bryce Elvin
2	Slade	Cameron	100703005	Cameron Slade
3	Sprung	Lucas	100699915	Lucas Sprung

Table of Contents

Requirements

The project outline given acts as a customer for this project, and lists requirements that the project must satisfy. This project is to build a mechatronics system resembling that of a search and rescue robot meant to navigate its way through a mine in order to secure and retrieve a trapped or injured worker. The project is broken down into 3 phases. For Phase A, the robot will need to autonomously navigate the mine, represented by a maze, and successfully locate and secure the worker given their location.

Functional requirements for this project include the robot's ability to be fully autonomous, and cannot involve any human interference once it has entered the maze (with the exception of the addition of three "save moves" added in Phase B). The maze that it goes through will be of a known layout, with the worker at a known location. The layout of the maze given will include where the walls are located, as well as the starting location.

The robot's Physical requirements are that the design of the robot will have to fit within the size constraints of the maze, including a height constraint on the bottom section. The maze is built as a 4x6 grid of square cells on both top and bottom sections, and features a ramp that wraps around the grid to go from one floor to another.

The robot will have to make use of the kit provided by the course instructor. This kit includes 2 DC motors with built in encoders, a servo motor, 4 IR sensors, a selection of wires, 2 sets of wheels, a motor driver, and an Arduino Due. Additional components may be purchased for the project. This project will also make use of rapid prototyping methods, mostly 3D printing for construction of the robot.

In order to accomplish these tasks outlined by the customer, the robot will absolutely need to use the motors and the wheels to navigate the maze representing the mine. It will also need the IR sensors given for real time error scanning while the program is running and the robot is making its way through the maze. It will also need to have some method of securing the worker in the maze. Additionally, the IR sensors require a light to operate reliably, and this will be an added benefit to the miner, so that they can see the robot coming. Phase A only requires that the worker be secured by some means, retrieval and exiting the maze are not requirements for this phase of the project. The robot should be able to reach the cell the worker is in and stop there. The method of securing is not specified by the customer, and should be taken into

consideration when designing that the method of securing should be simple and effective, this does not need to be the most elaborate thing ever designed.

In Phase B, it is required that the robot is capable of completing Phase A, with the addition of returning with the miner to the starting location in the maze with a new time limit. The robot must be capable of bringing the worker back to the start of the maze within 3 minutes.

Background Research

In order to ensure the project requirements would be satisfied, background research into various components was required. The first main research question was the material in which the robot body would be constructed from. Factors such as strength and durability need to be considered as well as cost and weight. Constructing the body out of metal such as aluminium would be strong, durable, and lightweight however, the cost of aluminium would be too expensive for the purposes of this project, as well as time consuming to construct. A robot body made from plastic such as PLA, while not as strong or durable as a metal body, would be lightweight, cost much less and could be easily made with the assistance of a 3D printer.

After examining the provided kit which included motors, sensors and an arduino, looking up the data sheets for the provided components was required to determine how much of the components would be used. The provided motors are equipped with built-in encoders which can assist in measuring the speed and position of the robot [1]. The sensors provided are IR distance sensors which are highly accurate and can measure distance of up to 50 cm [2] which is more than enough for the project. The provided microcontroller was an Arduino Due which is best used for high performance prototyping [3]. This board operates at 3.3 volts compared to the 5 volts of other boards such as the Uno and Nano and is equipped with an ARM processor which is made with lower power consumption in mind, making it more ideal for smaller devices [4]. The board is also equipped with 12 PWM channels, 12 analog inputs, and 2 analog outputs which will assist in voltage regulation for the motors and sensors [3]. The motor shield that can attach to the arduino allows for the motors and sensors to be connected and powered as well as control the DC motors [5].

In order to power the microcontroller and attached components, a power source was needed. Power sources considered were a standard battery pack and lithium-ion batteries. One of the main differences of lithium ion batteries is the increased energy density which allows them to

deliver more power in comparison to other batteries [6]. This comes at the cost of these batteries' tendency to overheat as well as damage sustained at high voltage. A normal battery pack would not have high energy density but also would not have the risk of overheating.

Another task that needs to be fulfilled is the means to secure the miner. In Phase A, the minor only needed to be secured and in Phase B, the miner must be secured and transported back to the beginning of the maze. For Phase C, the miner must be retrieved and optionally retrieved. A gripper style design similar to Figure 1 would be able to retrieve the miner and keep them secure enough for transporting them back to the beginning of the maze. This method would require an additional electric source and means of actuating the gripper so it can open and close. This paired with the potential size of the gripper may cause other issues and if the gripper is misaligned, then the miner won't be secured properly. A scooper-bucket style much like what is in Figure 1 was also considered as it would provide a secure place for the miner to sit when retrieved. The potential downside to this method is if the approach of the bucket is off, then the miner won't sit right and could fall out before returning to the beginning of the maze which would not be acceptable. The size of the bucket would have to be constrained as the miner has to fit inside the bucket but the robot must still be able to traverse the maze freely. Double sided tape was considered for Phase A as a method to secure the miner, however with the increased requirements set out for Phase B, this option would not be reliable for keeping the miner secure when traveling back out of the maze. Phase C would allow for this to work but alternative options would be preferable.

Figure 1: Gripper and Bucket background reference

Source: Adapted from [7] [8]

Obstacles were an additional item to consider when traversing the maze as they are unexpected in location and if they cannot be dealt with, navigating the maze and securing the miner becomes a more difficult task. One option of dealing with the obstacles depicted in Figure 2 is with a sweeper arm to move the obstacles out of the way of the robot. The problem with this design is with the size of the obstacles and the size of a maze cell, this would only work if there was an opening in the maze to push the obstacle into which is not a guarantee. Another option shown in Figure 2 is to use an arm to lift the obstacles out of the way. This would be effective at clearing the way for the robot however, the lifting mechanism for the arm must be strong enough to lift the obstacle and hold it. A spatula type design would accomplish the task of moving the obstacles out of the way but may not be possible as there may not be enough clearance for the spatula to slide under the obstacle and might push it as a result. A means of removal of the obstacle from the apparatus to deal with the obstacles should be considered as to make the system be able to handle multiple obstacles which increases the overall robustness of the system.

Figure 2: Method of dealing with obstacles

Source: Adapted from [9] [10]

One of the core requirements of the project is the robot is autonomous and can navigate the maze by itself. Recursive backtracking is an algorithmic technique which tries to build a solution incrementally by removing failed attempts and keeping the correct path until it reaches the full solution [11]. This would build a full solution of the maze but due to the time constraints and size of the maze would potentially take too long. A maze algorithm that has the robot always turn right could assist in solving the maze, however the time constraint paired with the maze

solution potentially requiring more than just right turns would make this method inefficient. Dead-end filling is a maze solving solution which would involve taking the maze and filling in the deadends and paths leading to the deadends until there is one path [12]. The downside to this mapping algorithm is that in order to work effectively, there must be one defined solution which cannot be guaranteed. The shortest path algorithm tracks where each individual cell of the maze is in reference to the starting point until it reaches the finish then follows the shortest path of cells to the start [12]. This method would provide a clear path back to the beginning of the maze to bring the miner out of the maze but might take too long in the initial stage of getting to the miner. Hard coding the maze into the robot is also an option for navigation. With pre-existing knowledge of the maze layout, the best path could then be programmed and carried out by the robot. Code modules such as a cell length and a right or left turn could also be implemented to further assist the ease assuming these are constant. This method is valid however is dependent on the prior knowledge of the maze and would be unable to account for possible obstacles in the maze.

In order to get a better idea of how the robot would function, other existing robots were researched and analysed. One such robot is a Small Unmanned Ground Vehicle for Search and Rescue Operation by the European ICARUS project. This robot is equipped with systems to allow for maneuverability over unstructured environments such as collapsed buildings. Its limited size and weight prevents it from having sophisticated sensors and is operated by a tele-operator [13]. This robot is equipped for unstructured environments much like what would be seen in an actual mine collapse and the gripper that is equipped would be quite capable of securing the miner. The main issue is that its autonomous capabilities are very low due to the simple sensors it has equipped and at times only be semi-autonomous. Due to the mandatory autonomy for this project, that would be an aspect that can be improved upon.

Another Rescue Robot looked into was the Coal Mine Rescue Robot (CMRR). This type of robot, while as of July of 2020 is still in the prototyping stage, possesses most of what is required for this project. The CMRR is capable of traversing the unstable terrain of a mine and is capable of reconstructing the environment around it in 3D. This robot also implements binocular vision which is used to scan and create an accurate 3D image of the robot's surroundings [14]. This concept for the small scale tests is more advanced than necessary but would provide beneficial knowledge in a real world rescue.

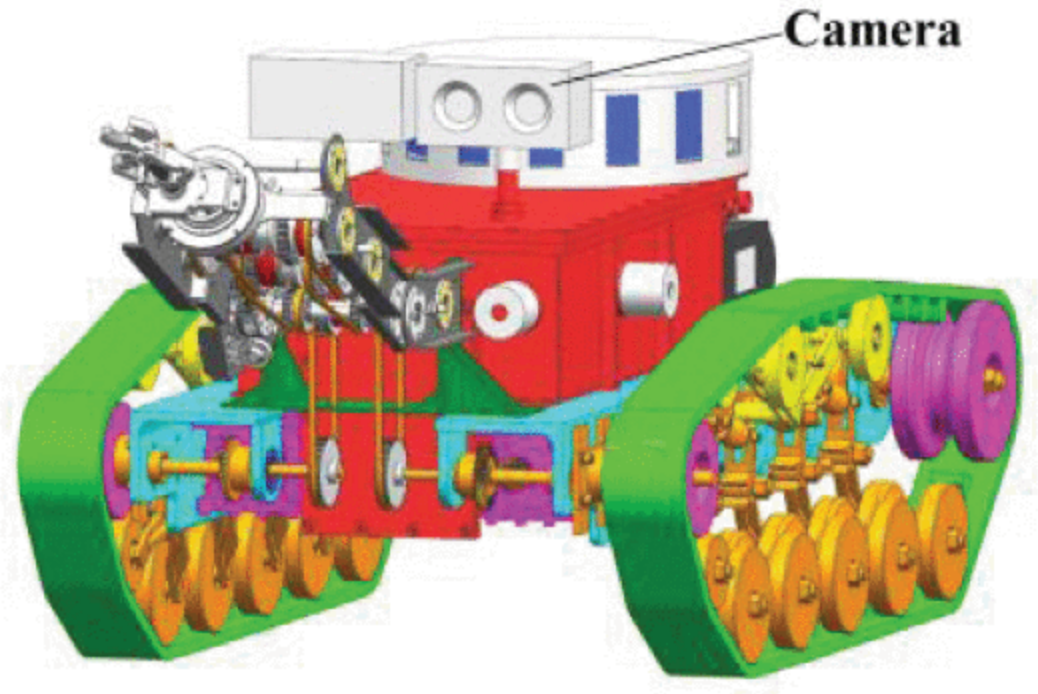
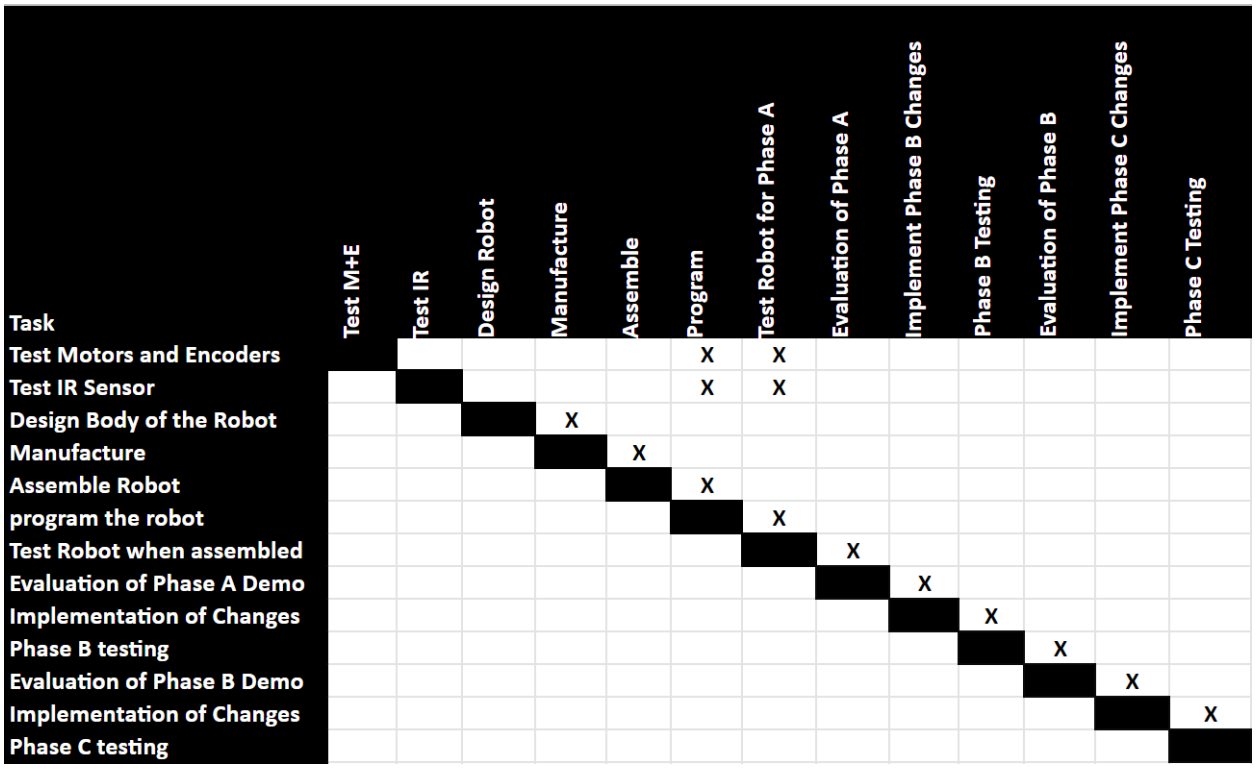


Figure 3: Coal Mine Rescue Robot (CMRR)
Source: Adapted from [14]

Project Plan/Management

Major tasks for this project were outlines and a timeline was created to manage project progress and resources. Before these tasks could have any deadlines associated with them, they key tasks had to be outlined first, and their connection to each other had to be determined. To do this a design structure matrix was constructed, which can be seen in figure 4.



Task	Test M+E	Test IR	Design Robot	Manufacture	Assemble	Program	Test Robot for Phase A	Evaluation of Phase A	Implement Phase B Changes	Phase B Testing	Evaluation of Phase B	Implement Phase C Changes	Phase C Testing
Test Motors and Encoders						X	X						
Test IR Sensor						X	X						
Design Body of the Robot				X									
Manufacture					X								
Assemble Robot						X							
program the robot							X						
Test Robot when assembled								X					
Evaluation of Phase A Demo									X				
Implementation of Changes										X			
Phase B testing											X		
Evaluation of Phase B Demo												X	
Implementation of Changes													X
Phase C testing													

Figure 4: Design Structure Matrix

The major tasks associated with the project were defined as testing the motors, encoders, and IR sensors to ensure that they are all functional. The design, manufacture, and assembly of the robot are also key milestones. Once all of the electronics have been tested and the robot assembled, then the programming of the robot and testing of the robot can begin. After that, evaluation of the design is done after the demo for each phase of the project, design changes are implemented, and testing begins again for the next phase of the project.

As can be seen from the matrix above, programming not only relies on the robot being assembled, but on the confirmation that the motors and the sensors are functioning properly. This way the design won't get to the programming stage before it is apparent that there isn't any working electronics. There are subtasks associated with testing the electronics including calibration and wiring diagrams. The design of the robot itself however does not rely on working electronics, only one the dimensions of things like the arduino and the motors. The electronics testing and the design of the robot can happen concurrently. Once everything has been assembled the programming of the robot's algorithm and the testing of the robot can begin.

The Design Structure Matrix also outlines the importance of the early stages of the project. Completion of the main robot and its program are crucial to the beginning of the project,

and have much more interrelated tasks. After the testing of Phase A concludes, the tasks become much more linear, as the evaluation leads to the changes that lead to the new testing.

Now that the key tasks have been identified, one can start to put them into a timeline. Creating a timeline also means prioritizing certain tasks over other ones. This project is split into 3 phases, A, B, and C. For clarity, 3 separate timelines were created for each phase of the project. These timelines are shown in the form of a Gantt chart in Figure 5.

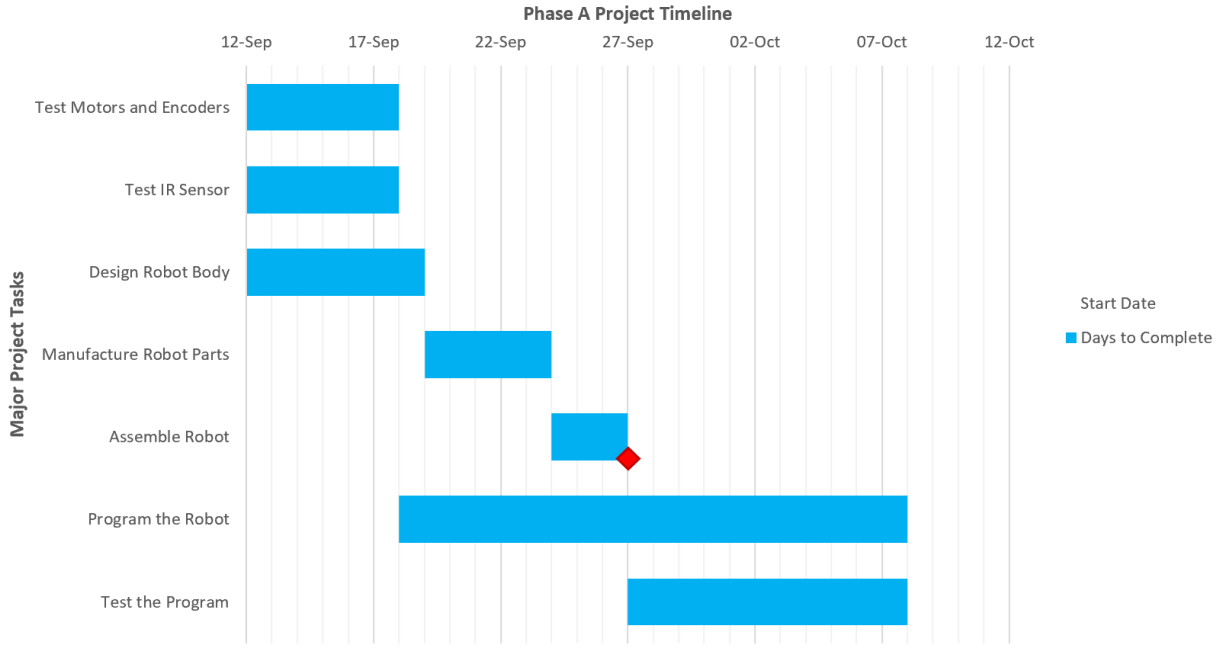


Figure 5: Phase A Gantt Chart

The blue Gantt chart takes the projects from the design structure matrix and converts them into a project timeline for Phase A. This helps further visualize what tasks can be worked on concurrently and when they need to be completed to stay on schedule. The first week of the project is dedicated to the design of the robot and testing of the electronics. The duration of these tasks lines up with the completion of the first lab, where one will have access to the maze for the first time. It is during the lab that any issues that come up during testing can be addressed, and measurements of the maze can be taken and then considered to either change or confirm the design of the robot. Once the design is finished, manufacturing can start for the parts for the robot. This task is given a week to complete, accounting for testing tolerances and any errors that may come up during the manufacturing process. The Timeline for manufacturing lines up with the second lab, where it will be verified how well the design fits within the maze. Once the parts

are manufactured, the assembly of the robot can begin. This task has the shortest timeline, as any errors in design or manufacturing should have been dealt with by this point already. The red diamond at the end of the assembly task marks a major milestone for the project. This is a firm deadline that should be met in order to allow as much time as possible for testing given the limited amount of time in the lab.

Programming the robot is given the most amount of time, as it will likely undergo more changes than any other task in the project. It starts when the electronics in the kit provided are confirmed to be working, and will continue to be optimized up until the demo of the robot on October 8. The testing of the robot itself is also given a significant amount of time, but is limited by the need for the robot to be fully assembled. While the sensors and motors can be programmed in an algorithm without the need for a full body, testing of that algorithm will require the robot to be assembled, hence the importance of the major milestone for the assembly.

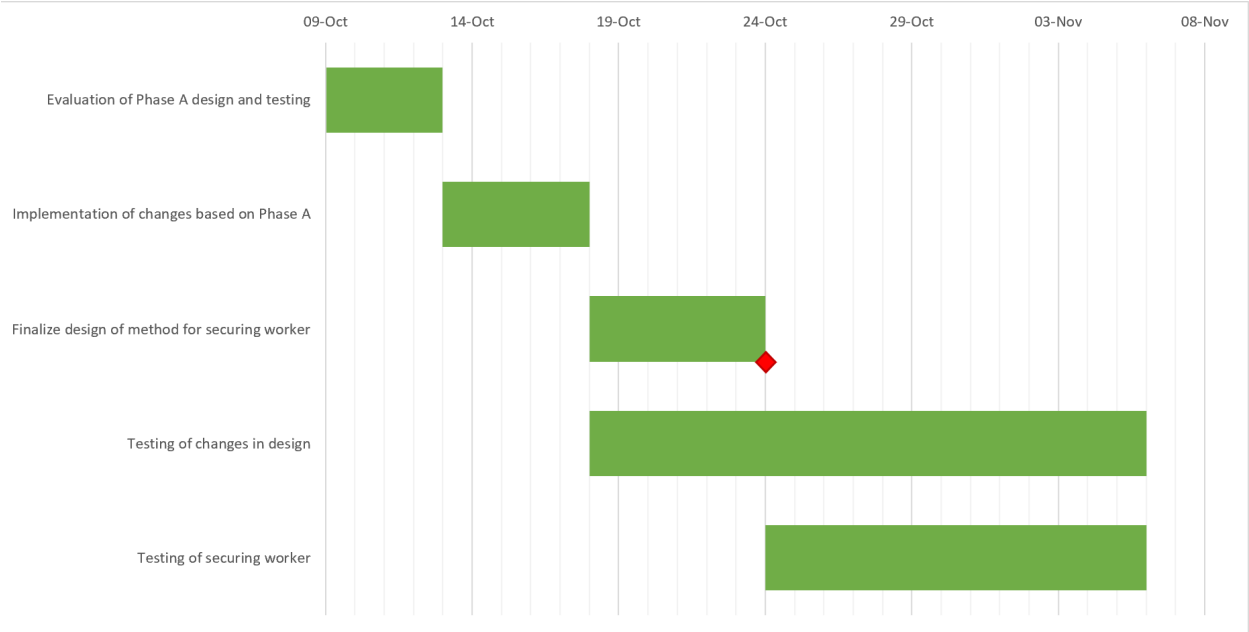


Figure 6: Phase B Gantt Chart

The green Gantt chart in Figure 6 shows the timeline for Phase B. It starts after the demo for Phase A, evaluating successes and failures of the demo and the testing leading up to it. Implementation of those changes is given a substantial amount of time, and then finalizing the design for the method of securing the worker is also given a decent amount of time. The design of the method for securing was done during initial concept generation, however it was not crucial to Phase A of the project. Phase B builds on Phase A, and the securing method will need to be

finalized for the demo of Phase B, hence the red marker indicating a major milestone. The testing of the changes in the design can begin once they have been implemented. Because the method of securing the worker was not focused on in the first phase, its design is not necessary for testing the changes to the Phase A design, these being changes to the movement of the robot. Once the design for the method of securing the worker is complete, the design can be updated and it can begin testing alongside the changes to the movement algorithm. This timeline culminates in the demo for Phase B.

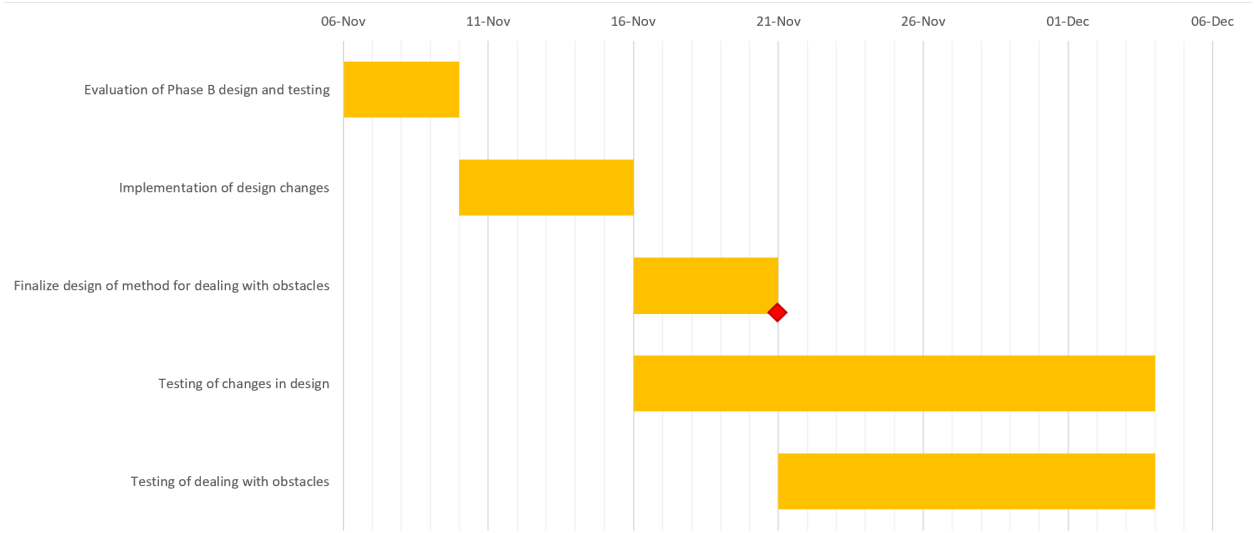


Figure 7: Phase C Gantt Chart

The yellow Gantt chart in Figure 7 shows the Phase C timeline. This timeline looks almost identical to the one for Phase B, consisting of evaluation/implementation of changes from the previous phase, and testing of those changes. Phase B also relied on finalizing the design of the method of securing the worker. Phase C however requires finalization of the design of dealing with obstacles in the maze. Like Phase B, Phase C can start testing its design changes from the previous phase as soon as they are ready, and the method for dealing with obstacles can be worked on while the testing is happening up until the point the robot is ready to be updated. There is another red marker on the finalization of dealing with obstacles and this is a crucial element of Phase C of the project.

From Looking at both the design structure matrix and the Gantt charts, it is observed that the Manufacturing of the parts and the testing of the sensors could create potential bottlenecks for the project. Most of these bottlenecks come from Phase A of the project during the robot's initial design. If the electronics are tested and there is an issue with one or more parts from the

kit, it will take time to fix or replace the components, which will in turn delay the task of programming the robot. This is largely why so much time is allotted for just making sure everything is working and why the task coincides with the time in the lab. Programming the algorithm for the robot also happens concurrently with the assembly of the parts. Manufacturing is also a potential issue if there are problems with tolerance, or issues with the time it takes to manufacture the parts. The best way to mitigate this is to have all the manufacturing done by us, rather than outsourcing the manufacturing to the school or another third party. This cuts down on time to travel to and from the school to pick up parts, and allows for the use of owned machines that are trusted to be more reliable than outsourcing. The school also has a time limit to how long parts can be manufactured for, and how large the parts can be. These are limitations that can be avoided by using in house machines. It is hoped that by working on the design and electronics concurrently, the time before the presentation can be maximized for testing before the date of the demonstration.

For Phase B and C, the testing of design changes from the previous phase is done at the same time as the final design for their respective additions to the robot to try and eliminate any potential setbacks during the design for the new additions. It is important to address the issues from the previous phase in order to ensure that they do not become issues during the next one.

Brainstorming

Phase A

During the initial brainstorming of ideas for the project for Phase A, it was decided that certain ground rules needed to be kept in mind for design. The most important one was centered around the design kit. The kit provided included an arduino, DC motors, and some IR sensors. While the design doesn't need to be limited to those components, the limited time constraints and budget for the project mean that it is more cost effective, in both time and money, to try and incorporate those components into the design. Choosing a different microcontroller would mean that one would need to find a way to interface them with the other components in the kit, or order new ones. Any new components would need to be ordered, which could create a delay depending on how long the parts take to arrive and based on the background research, the given parts are suited well for this type of project. Another rule focused on was centered around the requirements. Any design that is theorized needs to be capable of moving in a straight line, and

turning left and right. The design will also need to keep in mind the constraints of the maze itself. The robot needs to be able to fit inside a cell of the maze and to and within the height constraint on the bottom section of the maze.

The robot needs to be mobile, the easiest way to do that is with the wheels provided in the kit. The kit has two sets of wheels, one bigger, and one smaller set. However the design need not be limited with the wheels in the kit, however the wheels from the kit already have an interface with the rotating shaft of the DC motors. However there are only 2 motors in the kit, if the robot would use 4 wheels, the other 2 would not be constrained by interface with the motors and would be able to spin freely. 4 wheels would provide a stable mode of transportation, and could be connected to a main body to hold the arduino.

In addition to the 4 wheel design, a 3 wheel design was also considered, Using 2 motors to drive the robot, and a 3rd wheel that spins freely to support the rest of the robot. However, this design creates a potential problem when considering how the robot will turn. If the robot turns like a conventional vehicle, then the wheel would be fine, but then there's an issue with needing to initiate a turn before the robot enters the cell with the turn. This issue is resolved by having the robot pivot rather than turn, however creates an issue with the back wheel dragging as the robot turns. A potential way to solve this problem would be to create a roller ball joint rather than a wheel. By holding the ball inside of a cage, and fixing it to the end of the robot, the ball would be able to roll with 2 degrees of freedom solving the issue of the wheel dragging.

The robot also needs some form of securing the worker in the maze. The first thing that comes to mind is to use some sort of gripper arm, to grab the worker. As researched, this would require a mechanical mechanism to grab the worker, and would then need some sort of electrical method of triggering the mechanism. This, while a reliable solution, may unnecessarily overcomplicate the system for securing the worker. Another solution researched and what might be easier to implement is some sort of bucket, similar to what you would see on an excavator, to scoop the worker, rather than grabbing them. Some kind of adhesive could also be considered, but that creates an issue in the future with releasing the worker.

Depending on how the electronics are oriented will also affect how the sensors are placed on the robot. To avoid any conflict with the microcontroller the sensors could be placed on the bottom of the robot, but there could potentially be an issue with space on the underside of the robot with the motors and wheels.

Initial thought was also given to the algorithm that the robot would use to navigate the maze. The maze will be of a known layout, however the layout will not be provided any sooner than 48 hours before the official demo. Initial ideas were to use the IR sensors to determine where the robot was and have it be the primary communicator to the motors. The reading from the sensor would have the robot either move or not move. This however creates a potential issue when turning, as it is impossible to know where all the walls will be, and determining where to turn would be an issue. Focus then shifts to using the encoders on the DC motors. Since the maze uses a grid, if the size of each cell in the grid is known, the encoders can be used to track the distance the robot has moved, and then all that would need to be done is to program the distance of each movement once the layout is released. Potential issues with this are the fact that the distance would need to be measured precisely every time. If there was any error in the dimensions of the maze, the robot wouldn't be exactly where it needed to be.

Iterating on the last idea and paired with research into various different algorithms, it was theorized that the program could have a function for each movement. Have a function for “move forward one cell” and one for “turn θ radians”. Then, with the layout of the maze, the functions can be put in the order for the layout of the maze, and the robot will execute them. The issue of the exact distance was again looked at and a considered solution was to use the IR sensor as an error check. If the distance was programmed, and it is undershot, the robot would hit the wall and not make the turn. However, it is possible to overshoot the distance on purpose, and use the IR sensor to tell the robot if it is too close to the wall in front of it, and then to turn before there isn't enough clearance between the wall and the robot to make the turn.

Phase B

For Phase B, additional focus was given to the method of securing the worker. The first things that came to mind were things like electronic grippers and arms that could extend and secure the worker. Anything that moved would likely need to have additional electronics on it, like servo motors to control the actuation of the gripper. Alternatively, a bucket or scoop could work to secure the worker on a platform, and carry them out of the maze. Thought was also given to types of adhesives that could be used separately or in addition to the methods mentioned above. Double sided tape or something similar could be put on the robot to help them keep the worker secure when in motion. Thought was also given to the gripper in the context of a full

scale rescue bot in an actual mine. While not practical for our scale model in the maze, if a kind of scoop/bucket was used in an actual mine, it could be used to dig out a section of the ground around the worker, similar to how a bulldozer bucket would function, and then instead of just carrying the worker, it would carry the section of the ground the worker was sitting on. This would remove the need to touch the worker in any way, which would be especially important if the worker was injured in such a way where moving them would cause further damage, for example, broken bones or spinal injuries.

Along with the method for securing the worker, additional brainstorming was done based on the performance of the design during the previous phase. The idea of using additional sensors on the side of the robot to help combat drifting from left to right was one that came up many times during evaluation of the test results from Phase A. Small adjustments to the arduino mount and motor bracket were also brainstormed. Adding holes for screws to secure the motor brackets instead of a less mechanical solution as used in the previous section was also something we wanted to add for the Phase B design.

The rear support was also something that was readdressed during the brainstorming part of Phase B. The ball-socket support got stuck on a gap in the floor, and the robot was unable to continue through the maze. Previous ideas were revisited, like adding a wheel to the back that would turn. The wheel design was also iterated on, using a plastic gear/sprocket instead of a rubber tire to avoid slipping on the ground, but would still allow the wheel to turn to climb small obstacles. The ball-socket support was also “evolved”, consideration was given to a fixed support with a convex shape on the bottom that could allow the robot to overcome bigger gaps in the floor, as well as minimize surface contact between the floor and the robot, reducing the friction as much as possible.

The basis of the control algorithm was also readdressed during the brainstorming process of Phase B. Originally, the only feedback control into the system was the front IR sensor, which would detect proximity to a wall in order to prevent running into an obstacle. However, upon further testing, it was decided that it would be necessary to have additional sensors, in order to prevent a collision with side walls as well. To do this, a PID control system could be implemented that can be set to track either side wall (or neither wall). This would be greatly beneficial to traversing the inclined surface of the ramp without having to worry about errors present in the incline. Another control system would be to instead use both side sensors

simultaneously, and attempt to use a PID to keep the difference in distance between the two sides as close to zero as is possible. This method would provide more robustness based on the lighting conditions of the maze, however, it also requires that there be a wall on either side of the robot at all times in order to function properly, which simply may not be possible, and cannot be relied upon. This means that the PID should have some means of disabling/enabling the path correction system if necessary.

Finally, in Phase B brainstorming, there was an error realized with the IR sensors, in which different lighting conditions were tripping sensors too early. Some ideas to solve this issue were devised, such as an LED on the front of the robot, attaching a separately powered flashlight, or other means of increasing the brightness of the front of the robot so it can be kept more consistent.

Phase C

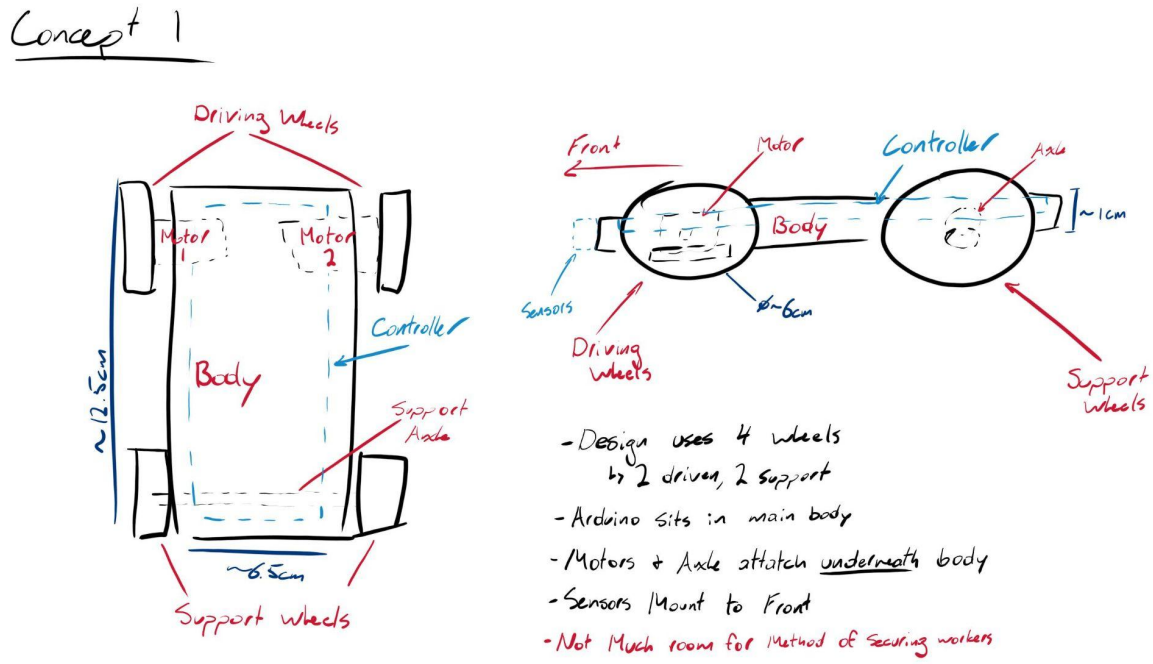
Phase C brainstorming revolved around how to deal with the obstacles that would be added to the maze. Many ideas and concepts were generated. Initial ideas involved housing servo motors and an actuated sweep arm to clear the obstacles out of the way. Using this windshield wiper motion we would be able to push obstacles clear of the path. Building on this, an axle could be attached to the front wheels of the robot. This axle would have a cam/follower system on the front end, enabling for an entirely mechanical solution to pushing obstacles along the path. Varying the end tool of the follower could allow the robot to direct the obstacles to the sides of the maze and out of the way.

A different approach to the obstacles is to lift them rather than pushing them. First drafts of this idea used a spatula type tool that could be shoved under the blocks, and then lifted quickly to throw them out of the way. An actuated gripper similar to ideas for securing the worker, but on a larger scale could be used to lift blocks, similar to a garbage truck type mechanism. Adhesives attached to an actuated arm would replace a gripper to avoid having to put more electronics on the end of the arm. Because phase C only required us to go in one direction, and not return to the beginning of the maze, if the robot dumped the obstacle behind it, it would not matter where it landed. Some ideas, largely inspired by booster rockets on space shuttles, would have the robot grab the obstacle, and then have the actuated arm detach from the main body, removing the need for the robot to let go of the blocks.

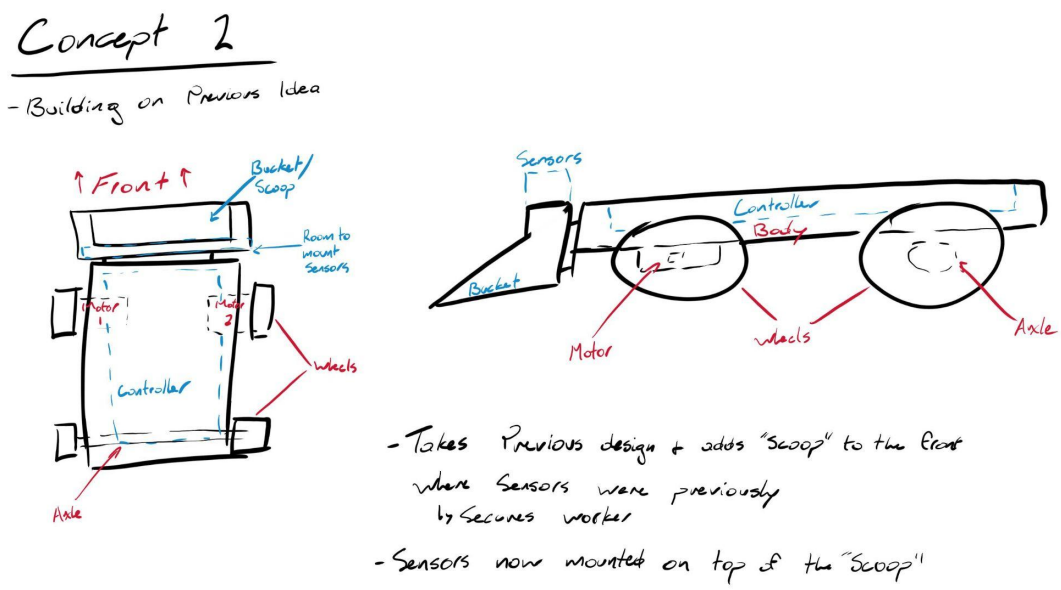
Other, less practical ideas were thought of, that might have more application if this were a real search and rescue bot, but may cause complications with the prototype maze. These ideas included using small scale high power lasers, or small blow torches which could be used to melt the obstacle so that it was no longer in the way. Even more impractical ideas involved flooding the maze to float the obstacles out of the way.

Sketching Ideas/Concept Development and Selection

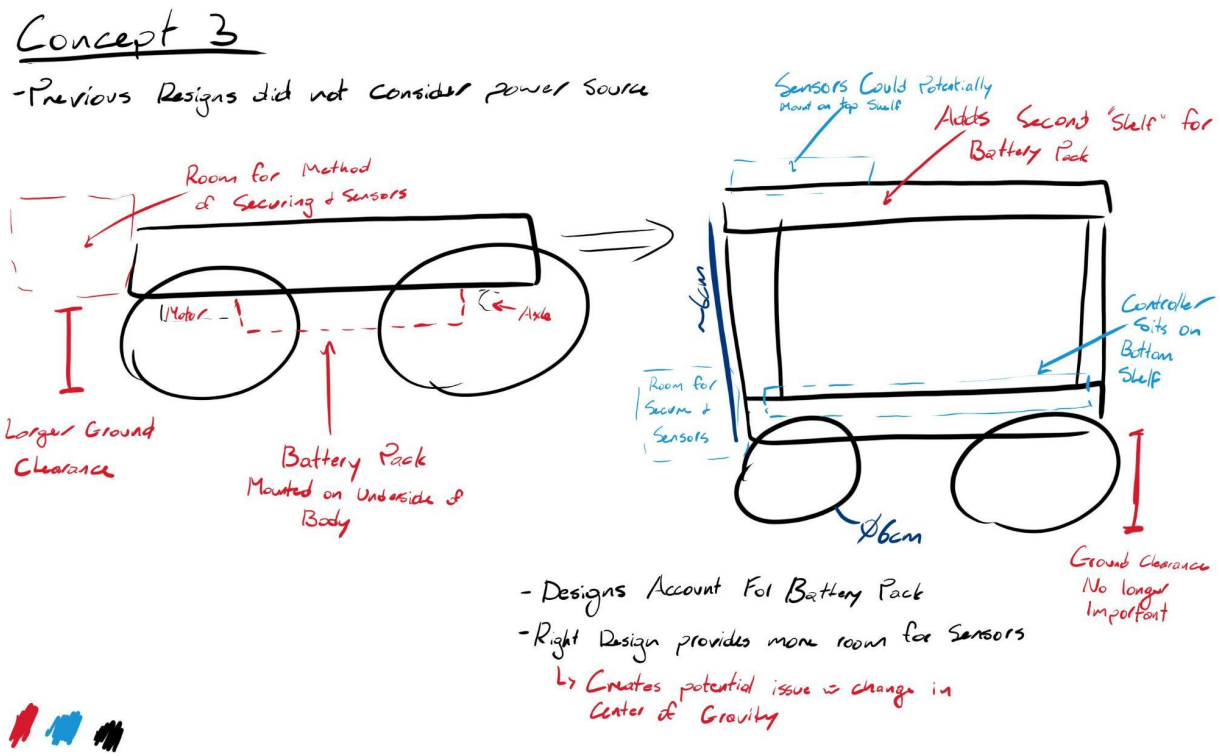
Concepts for the design of the robot iterated on each other as new information and ideas came to light. Each design sketch was labeled with important components and notes were taken on pros and cons of each design for clarity when comparing designs.



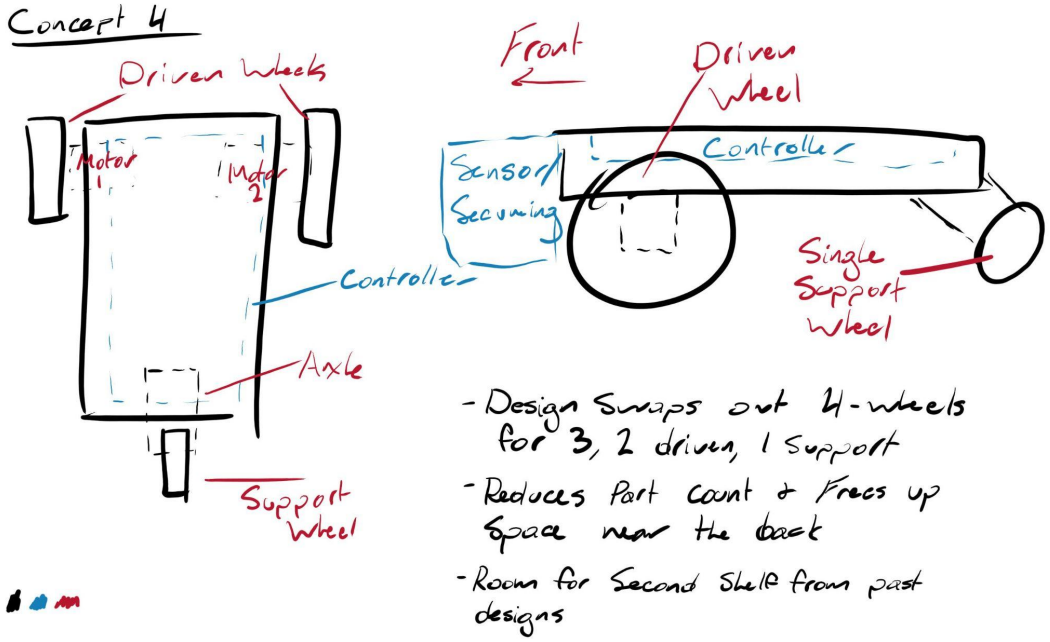
Concept 1 focuses on the 4 wheel design, and initial methods for connecting the arduino to the main body and motors.



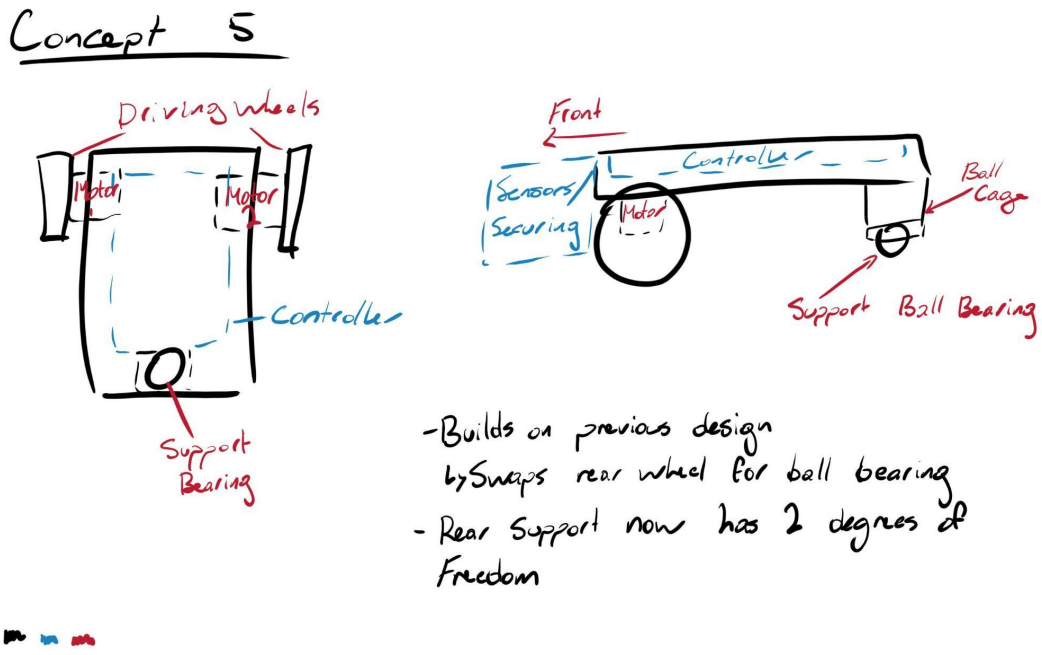
Concept 2 uses the same elements from the previous one, but considers more what the method of securing the worker will be and where it can be placed.



Concept 3 explores a more vertical style of design making room for more electronics. It makes use of a second shelf, however plays with the center of gravity and the amount of manufactured parts of the design.



Concept 4 shifts to a 2 wheel design with a supporting wheel on the back. Room for electronics and method of securing the worker can be taken from other concepts



Concept 5 builds on the previous design, addressing the issue of parts dragging while in motion, specifically during turns.

Concept generation for Phase B was based on the performance of the robot during Phase A. Since the overall body of the robot didn't seem to be an issue, it was decided that any additional features for Phase B should fit onto the Phase A design. Certain elements of the Phase A design were kept, these being the Arduino mount, and the motor brackets for the wheels. Phase B concept generation started with addressing any changes that would be made to these parts.

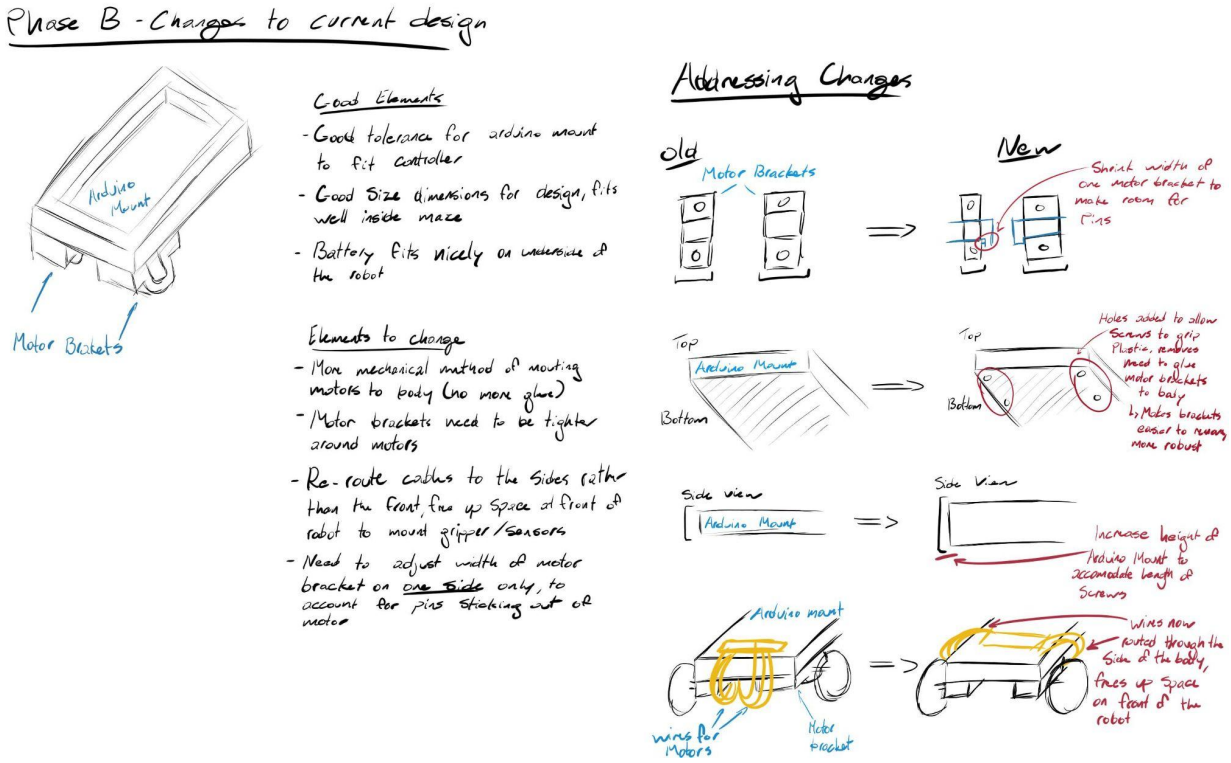


Figure 8

As described in Figure 8, Only small changes were made to the arduino mount and motor brackets. This portion of the concept development started with evaluating what was good about the old design, and what elements needed to change. This list of changes was then drawn out, showing the old design vs the new one for each change that was made to accommodate elements in the design. This drawing illustrates shortening one of the motor brackets to allow room for some pins on one of the motors, adding holes to the underside of the arduino mount so that they motor brackets can be screwed in instead of glued to the bottom, increasing the height of the

arduino mount to account for the length of the screws, and rerouting the cables to the sides of the robot clearing up space on the front. An element that is listed but not drawn out is tightening the tolerances on the motor brackets to reduce any wiggle room the motors may have during operation. The rerouting of the cables to the sides allows for room to mount the method for securing the worker to the front of the robot, as well as clearing up space for the sensors.

After changes to the parts that were being kept were sketched out, concepts on the parts being added and changed were created. These concepts started with the redesign of the rear support. These concepts can be seen below in figure P

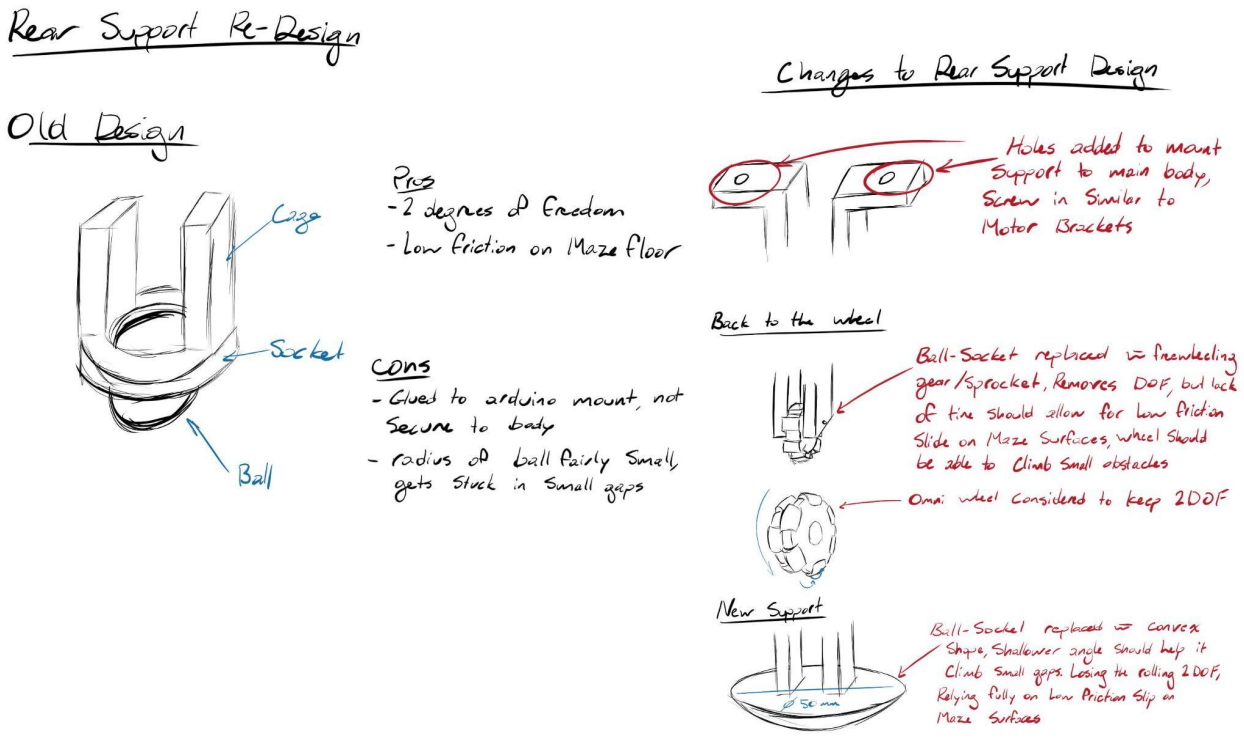


Figure 9

As the sketches in Figure 9 describe, concepts were created to replace the ball-socket roller that was used for Phase A. First off, the interface between the support and the body was redesigned so that it could have a more mechanical method of connecting to the arduino mount. Very similar to the motor brackets, holes were added to allow screws to fix the support in place. From there, ideas involving different types of wheels were visualized, including a rotating sprocket, and an omni wheel to try and maintain the same degrees of freedom that the ball-socket support had. In addition to the wheels, a new design was sketched out as a way to try and evolve the ball socket support. We couldn't add a bigger ball, as there wouldn't be enough room, but we

could make a CAD model with a high diameter, and cut it so that it would fit the robot size. The shallow angle would allow it to "climb" over small obstacles like the gap that the robot got stuck on during Phase A.

The major concepts developed during Phase B are the method to secure the worker. During this portion of the design, it was heavily debated whether or not more electronics should be added to the robot. Adding any major electronic components to actuate a gripper would conflict with the initial design decisions in Phase A to keep the robot small and compact. It would also further complicate the code, which would take up more processing power from the controller. On the other hand, a purely mechanical design might not be able to reliably secure the worker. A few different ideas were sketched out and can be seen in Figure 10

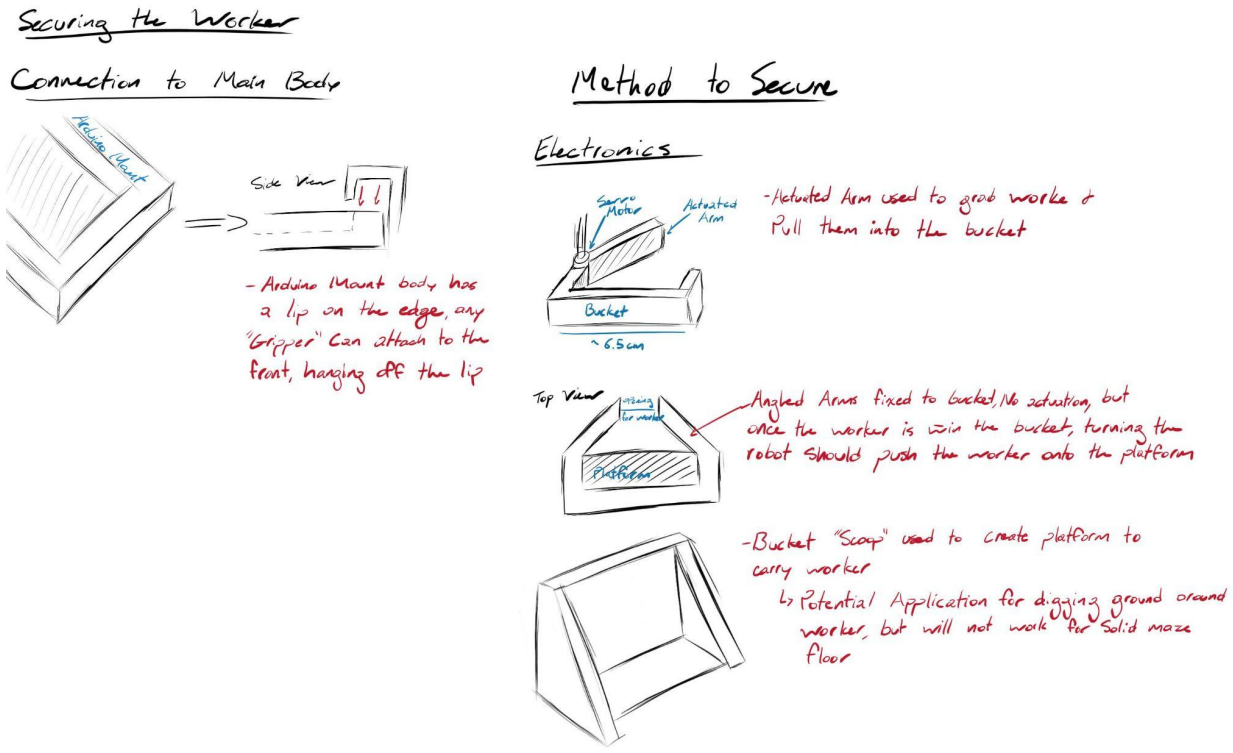


Figure 10

The first thing that was visualized was the way that the "gripper" would mount onto the main body. Looking at the current design, it was decided that the easiest way to mount anything to the front of the robot would be to use the lip that runs along the edges of the arduino mount as a way to hook on the gripper. From there, different versions of the gripper were drawn out. The first included an actuated arm with a servo motor, used to pull the worker into a bucket. The

other two concepts were purely mechanical, and would use the motion of the robot to secure the worker on the platform.

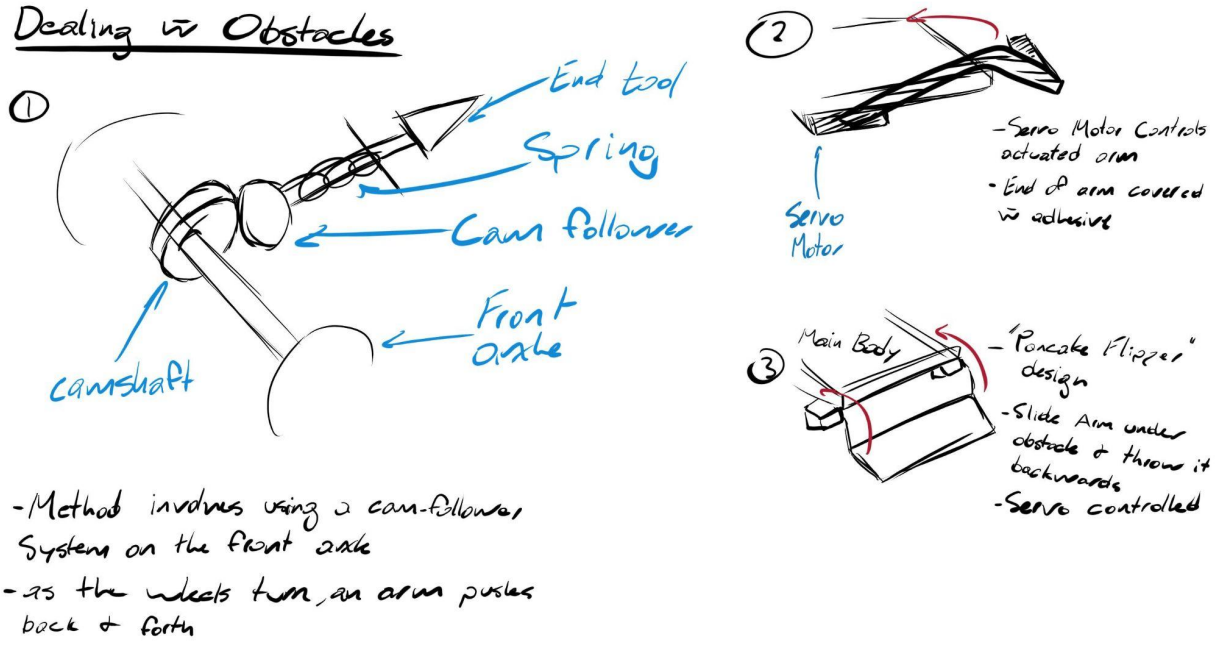


Figure 11

Some of the methods of dealing with the obstacles from the brainstorming phase were sketched out in figure 10. The first concept involves the cam-follower system, using a spring to control the return motion of the actuated arm on the front end of the robot. This would be an entirely mechanical system. Concepts 2 and 3 both involve servo motors for actuation. One involves an actuated arm to reach down and pick up the obstacle with an adhesive, and the other uses a spatula type design to scrape off and throw the obstacle out of the way.

Final Design

Evaluating all the concepts above from their test results of phase A and phase B, New concepts were chosen, and some old ones were swapped out. For the body of the robot, the final phase C design uses concept 4 rather than concept 5, which was used in both phase A and B.

For the method of securing the worker, it was decided that an entirely mechanical system would not be possible, and therefore the design was changed to use the servo motor and actuated arm concept that had been drawn out in early phase B brainstorming, seen in figure 10 .

For dealing with the obstacles, given the issues with the entirely mechanical method of securing the worker, the design opted to use one of the servo actuated concepts rather than the cam-follower system. It was decided to use the second concept designed in figure 11 to attempt to deal with the obstacles. The 3rd concept was considered, but ultimately passed over due to the expectation off issues with getting the spatula under the obstacle.

Functional Decomposition

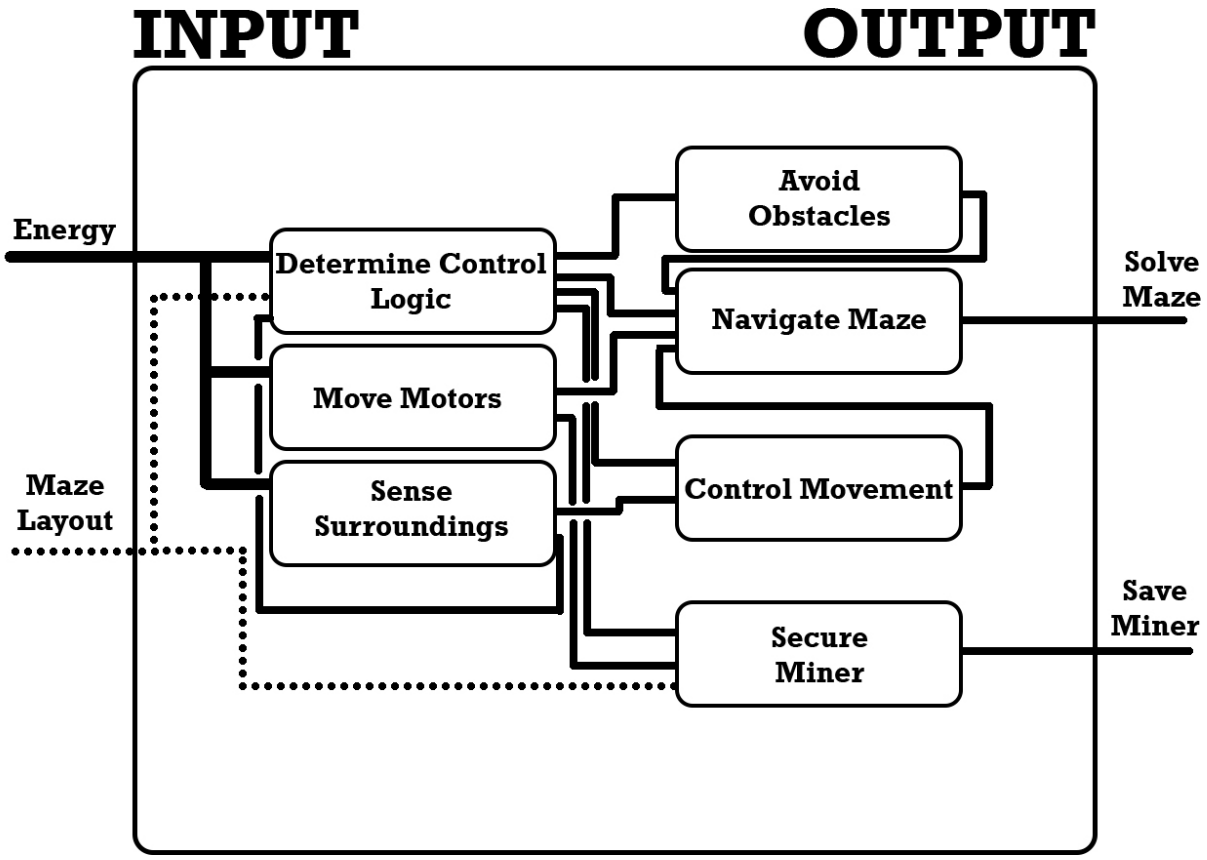


Figure 12: Phase A Block Diagram

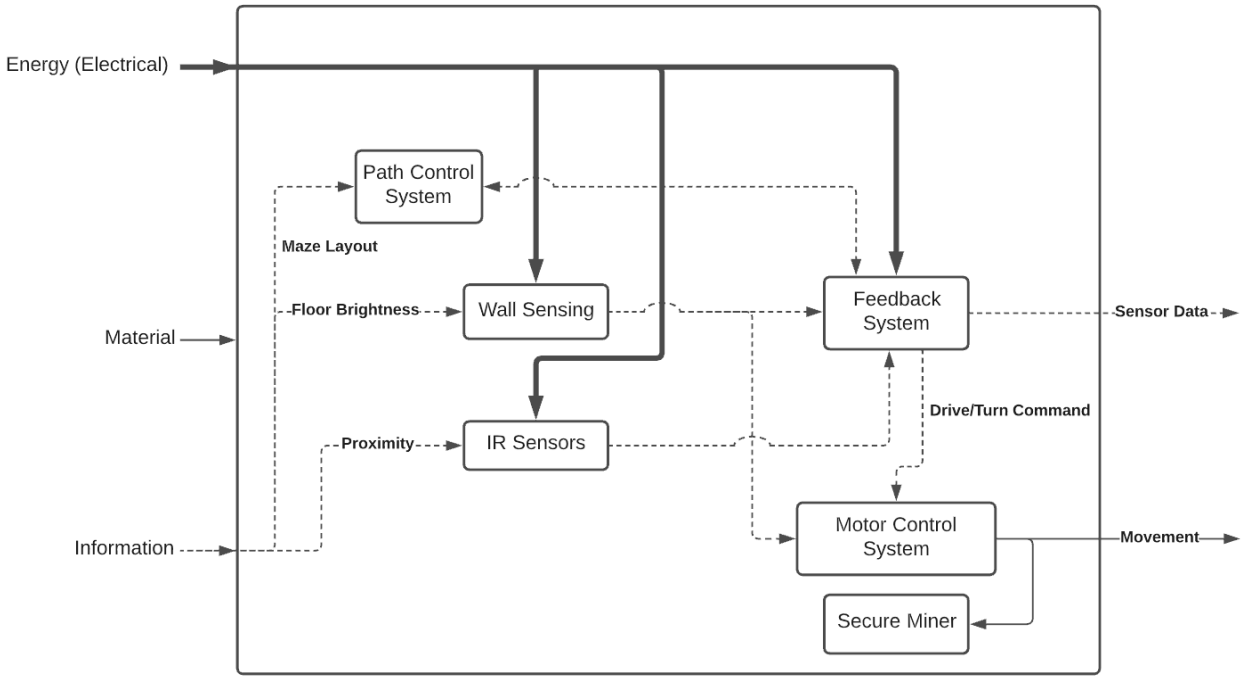


Figure 13: Phase B Block Diagram

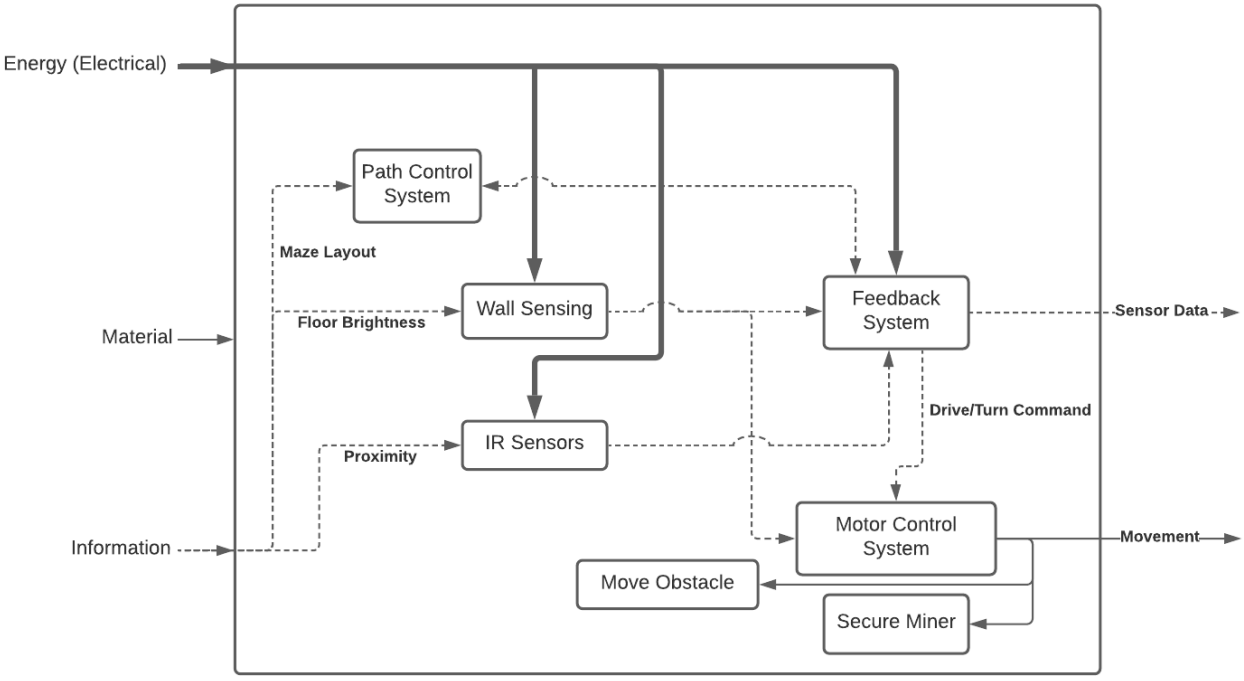


Figure 14: Phase C Block Diagram

As can be seen above, there is a functional decomposition that was used during the development of the robot. It is important as it keeps the project from becoming overengineered, and allows for the project to simply solve the problem it needs to solve robustly. The energy

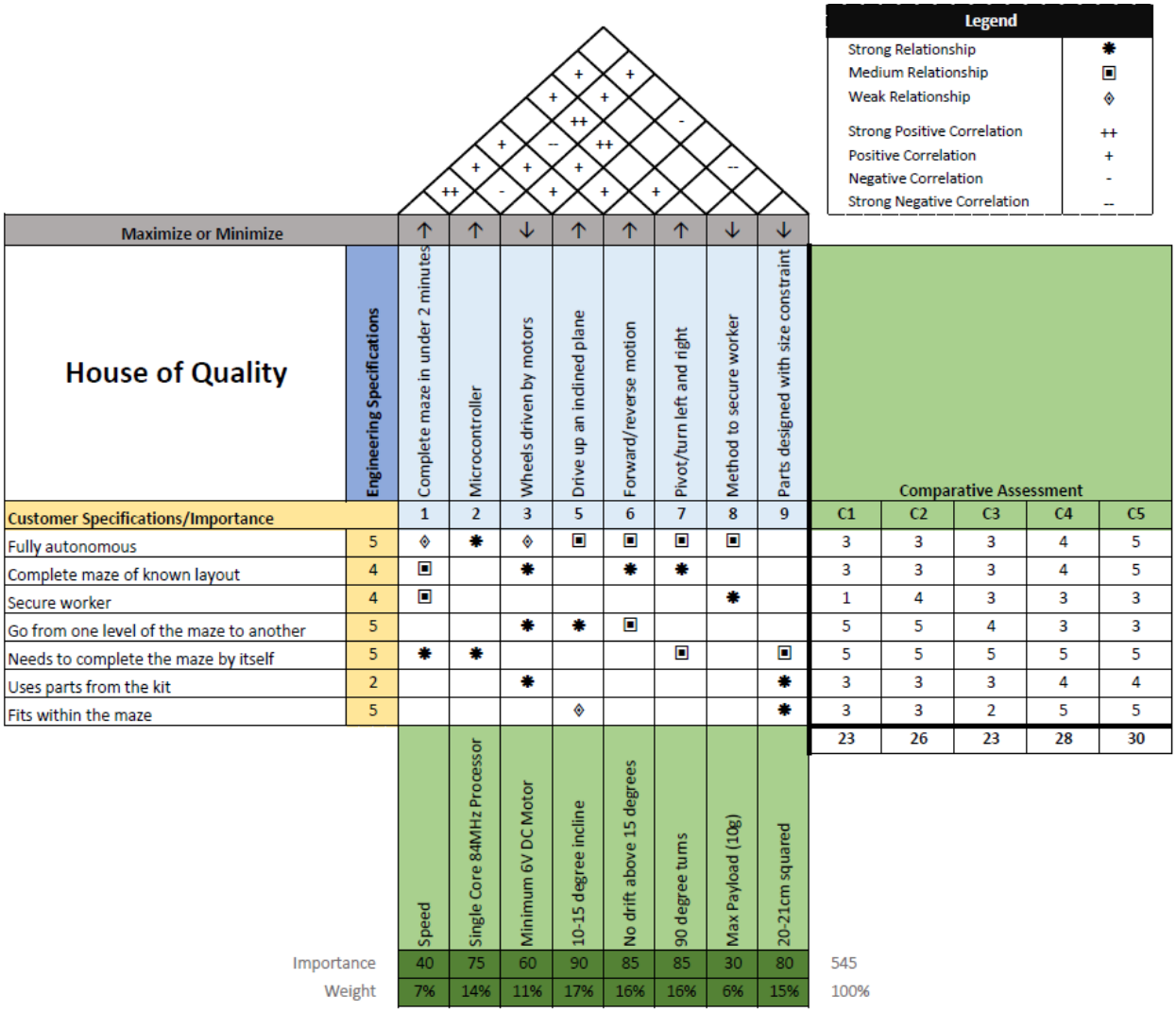
input to the system is simply the power input. In this case the power input is a 6V battery cell composed of four AA batteries.

The information provided to the system is the maze layout, the proximity information from the IR sensors, and the brightness of the floor (to tune the IR readings). The robot should be able to execute the rest of the program independently from the user. For this design, it is assumed that zero “save moves” will be required, and so they are not relied upon. There is no information output at the end of the system besides debug information during the design phase, since it is unnecessary once the miner has been saved. The physical output to the system is simply the movement of the robot towards the miner, and out of the maze. This is the only final output of the system.

Inside the functional decomposition, it can be seen that there is a feedback program. This feedback system is important, as it allows the IR sensors to send information back to the control system and change the path system during operation. This allows the robot to avoid travelling too far or not far enough, and keep the robot from hitting a side wall or drifting by modifying the path variables, such as motor voltages and distance to travel.

Navigating the maze is the culmination of all of the other systems working together. Navigating the maze requires the control logic, the sensors to correct for error and avoid hitting a wall, and the motors to be run when necessary. Securing the miner will come from the motor control system. This uses the control system information in conjunction with the path control system to find the miner and secure him. This is only possible because the location of the miner is known. Without the location of the miner (which is supplied by the maze layout information) this would also require a sensor input to locate the miner.

Engineering Specifications



Legend	
Strong Relationship	✱
Medium Relationship	▣
Weak Relationship	◈
Strong Positive Correlation	++
Positive Correlation	+
Negative Correlation	-
Strong Negative Correlation	--

House of Quality

Maximize or Minimize		↑	↑	↓	↑	↑	↑	↓	↓					
	Engineering Specifications	Complete maze in under 2 minutes	Microcontroller	Wheels driven by motors	Drive up an inclined plane	Forward/reverse motion	Pivot/turn left and right	Method to secure worker	Parts designed with size constraint	Comparative Assessment				
Customer Specifications/Importance		1	2	3	5	6	7	8	9	C1	C2	C3	C4	C5
Fully autonomous	5	◈	✱	◈	▣	▣	▣	▣		3	3	3	4	5
Complete maze of known layout	4	▣		✱		✱	✱			3	3	3	4	5
Secure worker	4	▣						✱		1	4	3	3	3
Go from one level of the maze to another	5			✱	✱	▣				5	5	4	3	3
Needs to complete the maze by itself	5	✱	✱				▣		▣	5	5	5	5	5
Uses parts from the kit	2			✱					✱	3	3	3	4	4
Fits within the maze	5				◈				✱	3	3	2	5	5
		Speed	Single Core 84MHz Processor	Minimum 6V DC Motor	10-15 degree incline	No drift above 15 degrees	90 degree turns	Max Payload (10g)	20-21cm squared	23	26	23	28	30
Importance		40	75	60	90	85	85	30	80	545				
Weight		7%	14%	11%	17%	16%	16%	6%	15%	100%				

Figure 15: Phase A House of Quality

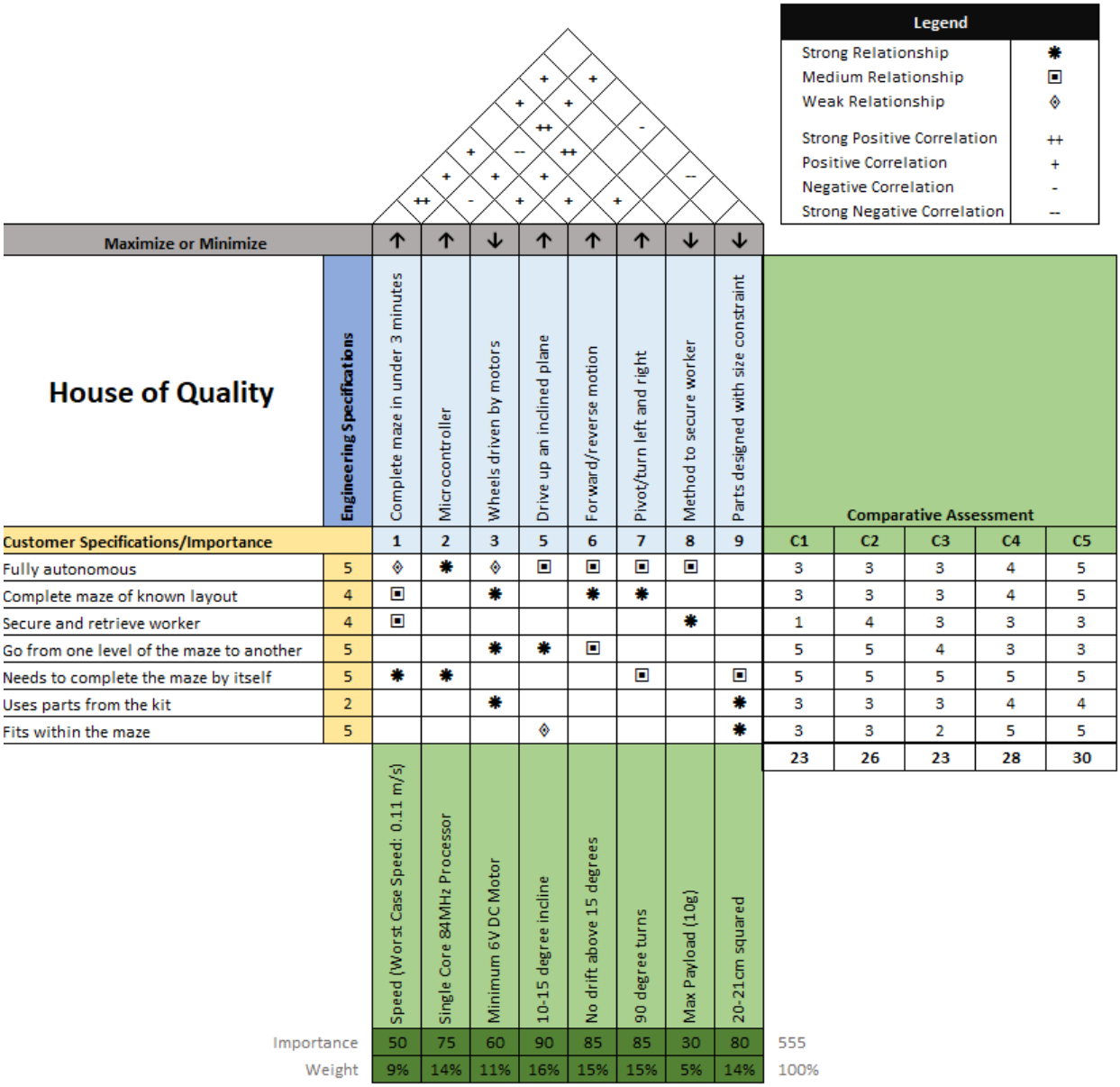


Figure 16: Phase B House of Quality

In order to determine specifications from the customer requirements, a house of quality was created for Phase A of the project, which can be seen in Figure 14. Specifications, certain components, and limitations were created for the functional requirements of the project. The main engineering requirements are the maximum size of the robot, which needs to fit within an approximate 21cm square. The robot will need to have enough power to climb an incline of about 12 degrees, and will need to travel straight without drifting, and make sharp 90 degree turns. The importance of each functional requirement was also determined. As can be seen, being fully autonomous was seen as the most important. The microcontroller, while important to being

fully autonomous, can be any type of microcontroller, the use of the arduino due is not as important to the project, and is reflected by the lower importance score in the house of quality.

The comparative assessment was done between the five different concepts created during the concept generation phase. Each concept was given a score from one to five based on how well it was thought each concept could achieve each task. Scores were added up and the concept with the highest score was considered as the best option to use for initial testing. Elements from other concepts can still be incorporated into the project to try and further improve each

A second updated house of quality was created for the Phase B portion of the project, seen in Figure 14. This New house of quality includes updated parameters to account for the changes to the project requirements for Phase B of the project. The time limit shifts from 2 minutes to 3 minutes, and now describes securing and retrieving the worker in the maze, not just securing.

Most of the other parameters remain the same, mainly those that describe the limitations of the robot, like the size, and the robots ability to navigate the maze. However the relationships between the specifications and customer requirements does not change much. The only meaningful parameter that needs to be considered as a larger engineering requirement for phase B is the time limit, and therefore the speed of the robot. Phase B will see the robot cover twice the distance, with only a 1.5x increase to the time limit.

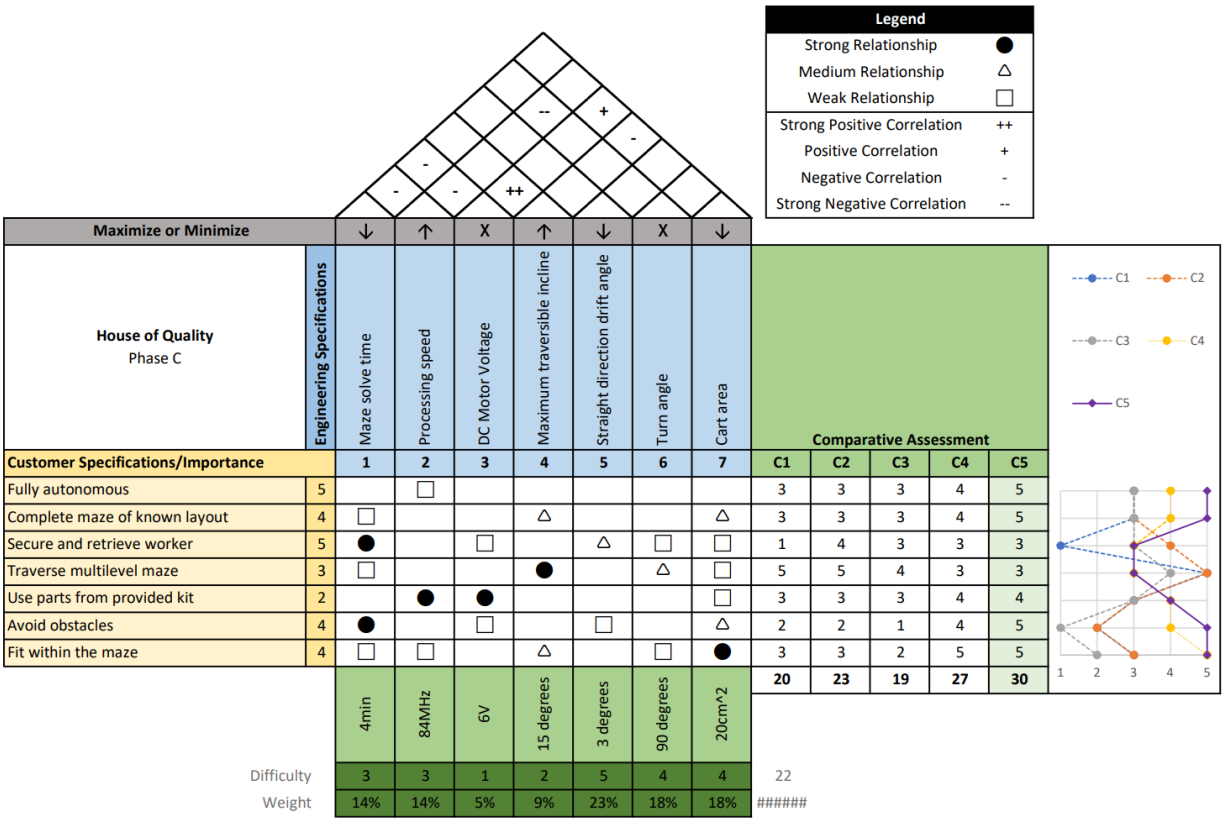


Figure 17: Phase C House of Quality

For phase C, some of the requirements were reworked. The addition of obstacles complicates the design process, and adds the difficulty of a new method to secure workers. For this particular new customer requirement, the assessment between concepts was done based on the ability to add an obstacle avoidance system to the robot. This also compensates for their respective size, as a large robot would have a harder time avoiding obstacles than a smaller one. For this requirement, both concepts 4 and 5 scored the highest, as they are easily modular and allow for simple attaching of an additional section to the main body. All other designs either require slight reworking to make this work, or they would need to be redesigned.

In some respects, Phase C was much more forgiving for time. This leniency is because there is an additional minute to complete the phase, however, the robot only has to get to the miner, and not return again. This means that the speed required is lower, allowing for more accuracy and safety. This means that processing power matters less to the phase, because the robot can simply operate slower with little repercussions. In addition to processing power, the robot also has the ability to move slower on straights and turns, allowing for a more accurate tracking of distance, and thus less drift from desired travel angles.

Form Design and Engineering Analysis

Primary constraint of the robot design is the cell size of the maze. With an approximate 21cm square, parts would need to be designed and oriented to allow for the robot to fit within the maze. A “Shelf” system was considered, but it was deemed that moving the center of gravity of the robot higher would cause potential issues when the robot needed to travel on an incline. The exact length of the robot was designed as 12.6 cm with a 10.7cm width including the wheels

For materials, the robot would need to be capable of carrying itself through the maze. Making the assembly as light as possible was a priority. Strength of the material does not play as much of a factor given the scale of the robot. PLA provides a lighter weight with a reasonable strength that should work perfectly for the project.

The main interface between the kit parts and any manufactured part would be the motors. The wheels were part of the kit, but the motors would have to be attached to the main assembly. This connection would have to be designed with the size of the motors, and how much the motors would move/vibrate during motion. Shock absorption was not deemed an important part of the motor/body connection as the maze has a flat surface all around. Any manufactured parts would need to be toleranced for the motors to ensure a secure connection to the main body of the robot. Lack of a secure connection will cause issues with how the software runs and the output of the control.

The robot was designed with the motors at the front end for better control when turning. Given the length of the robot, putting the driven wheels at the back would mean that the entire robot would likely not be in the cell when it turned, else the robot would not have enough room in the cell to make the turn. Having the robot as a front wheel drive means that it can do a full 90 degree turn while keeping both wheels within a maze cell.

For Phase B, the robot body did not undergo a major redesign. Instead, the bucket and sensors were designed to mount to the existing design. The cables for the motors were rerouted to the sides of the main body to clear up space on the front end of the design. The bucket for securing the worker was designed with a clamp that could hang off of the lip that goes around the edge of the main body. Tolerances for this were kept tight to ensure that the bucket would not fall off during motion or under the weight of the worker.

This phase also saw the addition of multiple sensors to the design. A front sensor was mounted to the bucket pointed forwards to sense for oncoming walls. This would allow the robot to make sure it had enough room between itself and the wall in front of it to make turns reliably. Two sensors were mounted to the sides of the bucket in front of the wheels. These sensors are used for the PID controller as the robot moves down the ramp. If the sensors were mounted behind the wheels, the robot would encounter an issue where both sensors would get further away from the walls on either side as it turns. Keeping the sensors in front of the wheels allows the controller to work reliably and as intended.

Design for Manufacturing

Multiple team members on the project have access to a 3D printer, so it was decided that 3D printing would be used as the main method for manufacturing parts. It was also decided that for this project, it is desirable to manufacture as little parts as necessary for the design. Design for 3D printing on an FDM printer meant that while there was the freedom to design whatever is needed, there would be certain aspects of the parts that would need to be kept in mind.

Parts would need to minimize the amount of overhang in order to keep the amount of support material low, and increase the overall quality of the printed parts. Parts would also need to be toleranced in accordance with the accuracy of the printer. FDM printers, especially the ones available for manufacture, usually require an extra millimeter or two to account for how the part is oriented and any accuracy errors in the printing process. This accuracy error would need to be kept in mind not just for parts interfacing with the electronics in the kit, but for connections and mechanisms between two or more 3D printed parts.

Parts would need to be designed so that they could be oriented on the build surface with a significant amount of contract with the print bed. Smaller surface contact with the build plate can cause adhesion issues when the print starts to get towards the top layers.

Parts were designed to minimize vertical distance when printing. If the vertical distance of a part could not be avoided, then the part was oriented in such a way that the vertical distance was angled either horizontally or at an angle from the build plate. Large vertical distances on an FDM printer are susceptible to layer shifts, which can cause issues with the strength of the part.

Parts were designed to minimize sharp corners. Shark 90 degree angles can cause layer shifts in the print if the printer is not stable and on a sturdy surface. If sharp corners cannot be

avoided, parts were designed with fillets to round the corners to make the print smoother and more efficient for the printer.

The design process for this project also sought to minimize the amount of manufactured parts as much as possible. The less parts that had to be manufactured, the less time would be spent waiting on print times, and mitigate the potential bottleneck in the manufacturing process. Using mostly off the shelf parts like the ones included in the kit would also help to minimize any tolerancing errors.

Design for Safety

Early in the design process, a list of possible risks and their respective likelihood and consequence were evaluated. The scale chosen was a 1 - 5 scale, with 5 being a fundamental risk, and 1 being a low risk. The following table is a description of the risks associated with the operation of the robot.

#	Event Description	Likelihood	Consequence	RPN
1	Loss of power	1.8	5.0	9
2	Impassable terrain	1.4	4.8	6.72
3	Sensor malfunction	2.6	2.2	5.72
4	Circuit disconnection	2.3	3.6	8.28
5	Software error	1.8	3.8	6.84
6	Motor encoder malfunction	2.2	1.0	2.2
7	Assembly damage	1.0	3.7	3.7

On this particular scale, it was decided for this design that an RPN below 4 would be a low risk, below 9 would be a medium risk, below 15 would be a high risk, and anything above that is a fundamental risk. The risks were then plotted as shown below.

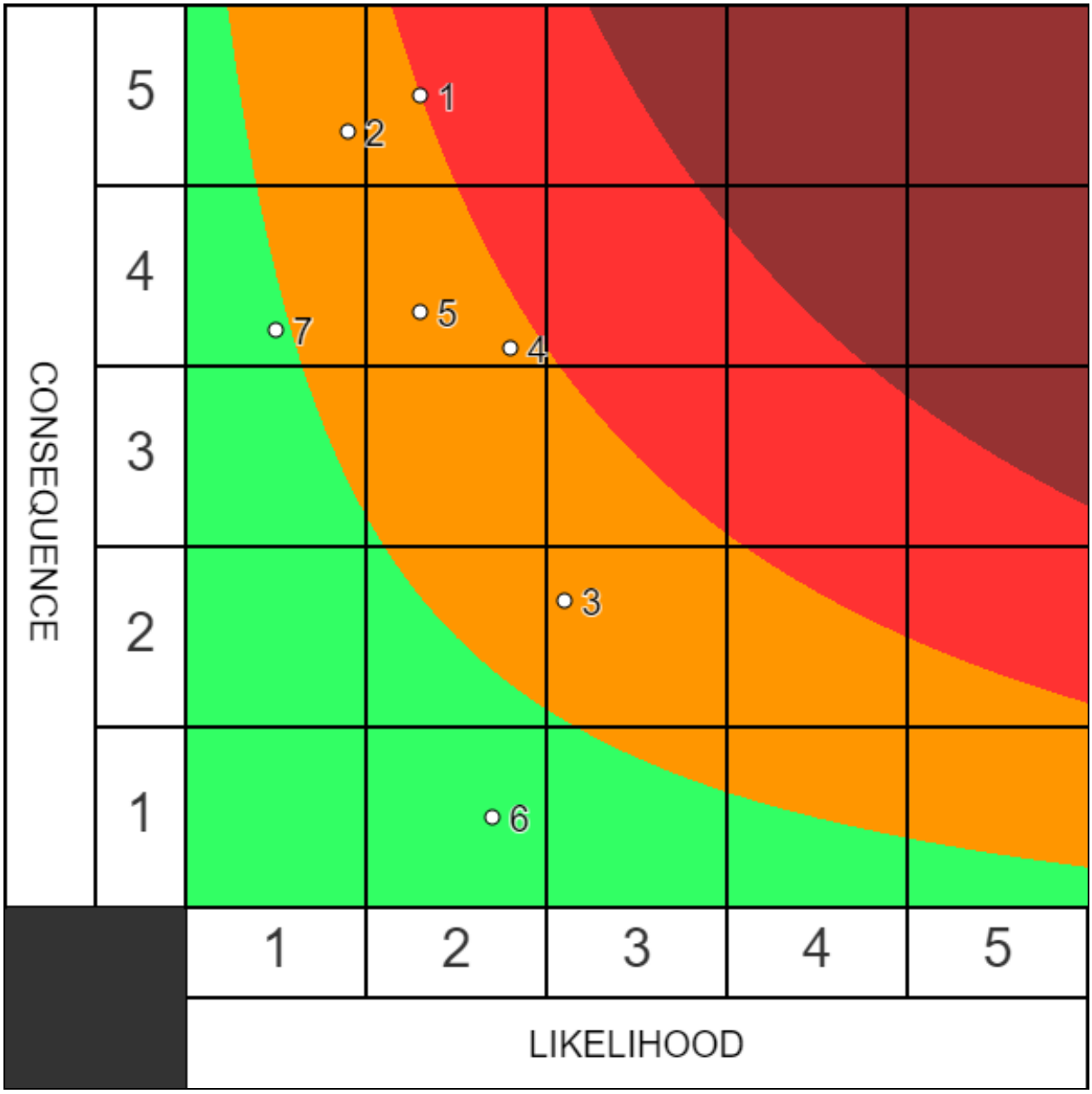


Figure 18: Risk Plot

In Phase A, the most fundamental risk to the operation of the robot was risk 1. This risk was the risk of loss of power during operation. This risk is no longer a fundamental risk to operation, but still maintains its position as the highest risk present, being the only high severity risk. While the risk of impassable terrain was previously a high severity risk, changes to the design of the rear support to avoid gaps in the floor of the maze, and adding a "bang-bang" to compensate for the bars in the wall. This also includes the robot failing to make a turn, or hitting a wall and getting stuck on an object, however changes to the sensing capabilities of the robot

make this issue less likely. This has the consequence of possibly completely failing the rescue operation, and so is moderately severe, however, it is not as likely as the high risk. The remaining medium risks are placed where they are because they have either low likelihood, or low consequence. These risks are not guaranteed to result in failure of the operation, but could possibly lead to failure. Previously the risk of software failure was the only low risk, however, with the introduction of new sensors, and the implementation of a PID control system, this risk is now more likely, and could have a higher consequence, and so has been shifted to a medium risk. The only low risk is now the risk of motor encoder malfunction, which was previously a medium risk in design Phase A. It has been shifted lower since the implementation of the PID control system will compensate for errors with the encoders, and this risk is less likely/severe for that reason.

By using this analysis, attention can be given emphasis to the most severe risks of the project. For example, significant time is spent ensuring that the motors have the correct voltages to move up a slope, avoid slipping, and still not draw too much power from the Arduino or the sensors. Time is also spent verifying that high risks are mitigated as best as possible, such as experimentally measuring the voltages of the motors to get the robot to move in as straight a line as possible before the PID system has time to kick in. Medium severity risks are still given attention but receive less scrutiny and attention to detail than the other risks, in order to focus workload to fit the deadlines, and give the best chance of a successful operation.

Control Algorithm Design

For the control of the robot, it was clear that there would be the need for some way to measure distance travelled, radians turned, and also correct the course of the robot to account for any error inherent in the encoders. The code should be kept in separate, repeatable functions, so that the path can be changed quickly and easily. This means a function for straight travel distance, turn angle, and a way to measure the encoders independently (and simultaneously) from the main program.

Firstly, for the handling of the travel distance, the encoders will be manually measured. In the case of a straight path, only one of the motor encoders needs to be measured, which will reduce the load that the microcontroller needs to process at any given time. The way that the encoders are monitored is with an interrupt attached to encoder pin A on the motor. Whenever

the rising edge (or sudden voltage spike) is detected from the encoder pin, the program will then check the other encoder pin (B) to determine whether it is already high as well, or if it is still low. If it is low that means that the motor is turning counterclockwise, otherwise it is going clockwise. Now, using the number of encoder pulses per rotation, and the size of the wheel, these can be used to calculate the distance in centimetres. If the diameter of the wheel is 6 cm, and the number of encoder pulses per full rotation is 2940, then one can take the number of pulses, multiply by 6 * PI and then divide by 2940.0. Upon experimentation, this way of measuring straight distance travelled was found to be accurate within 2mm. There is a running tally kept of the number of pulses, and it is reset after checking the distance each time. For straight travel, only one of the motor encoders needs to be checked, so one is chosen arbitrarily.

For the handling of the turns, a more complex method would have to be developed. The method that was decided upon was a measure of the arc length. This would allow the program to reuse the straight travel distance logic, with a slightly different process. Firstly, instead of being given a distance in centimeters, the function will be given a turn angle in radians. Then, this value needs to be converted to a distance travelled. In the case of a turn, the arc length the wheel draws would be the travel distance. With measurement, the width of the cart from wheel to wheel is 10.7 cm. It was decided that the turns would be made by rotating a single wheel and keeping the other stationary. In this case, the arc length travelled would be the angle in radians, multiplied by the width of the cart. In order to find the distance travelled at any present moment, a running accumulator is kept of the current distance, using the same method as in the straight travel function. However, this time a motor encoder can not be arbitrarily decided upon, as one will be stationary. To solve this problem, one of the motor encoders is set to be disabled for the majority of the time, until it is required to make a turn in that direction. This allows for the speed of using just one encoder, with the benefits of measuring both when necessary. By keeping this as its own function, any turn can be programmed into the operation of the robot to solve the path.

One more issue with travelling straight that had to be addressed in the code was the distance travelled by a turn. When the robot turns on one wheel, its position is going to change. This change will be the width of the cart, since turning on one wheel 90 degrees will push the cart exactly its width in the original facing direction. This means that any straight travel that ends in a turn must be reduced by the width of the cart to compensate.

Finally, any error in the program operation to do with the encoders must be mitigated in some other way, independent of the motors. To do this, it was decided that multiple IR sensors would be used. The code will turn the IR sensors on in the setup, and create a timer interrupt that triggers every 50 milliseconds. This way, the sensors will not place too much load on the microcontroller, and will maintain a near constant time between checks, and will still function quickly. A function is created for each interrupt that simply sets a global variable to the reading from the IR sensors. Then, this can be used to verify the distance from the wall in front of the robot. By placing the IR sensor higher on the robot, it will identify incoming walls (and ignore obstacles and the miner). The reading from the sensor was found experimentally to be about 565 for the distance desired from the wall before turning. Therefore, the travel straight function simply needs to check not only for distance travelled, but also for the reading on the IR sensor. If the IR sensor reads a number above than 565, that means the robot is too close to the wall, and should stop. This will compensate for the small amount of error travelling with the motors in a straight direction, and will also reduce the likelihood of the robot crashing into a wall when turning.

In order to compensate for drift, an option is available for the travel straight function that allows the robot to track a wall on the right or left side to maintain a straight path. This utilizes an IR sensor on either side of the robot, and a PID controller. The PID controller output is then linearized with respect to the motor voltage and speed, using a square root function. Using this method, the slight horizontal incline of the ramps can be accounted for, and turns need less accuracy to avoid a failure. In addition the readings from these side sensors are routinely averaged using a moving average, in order to avoid the robot suddenly moving away from the wall when a wall stud is present. Below, an algorithm flowchart can be seen.

The final element of the design (which was incorporated in phase C) was the detection of obstacles and the method of moving them. This complicated the previous design as it requires a new attachment for the front to aid in securing the obstacle. In the program, the robot will always check for obstacles while travelling. It does this by accessing two front sensors and comparing the data. If the top sensor senses nothing, but the bottom sensor does, it means an obstacle blocks the path. If the obstacle is detected, the robot will execute the operation required to lift the obstacle. In this case, this is done by controlling a servo motor arm.

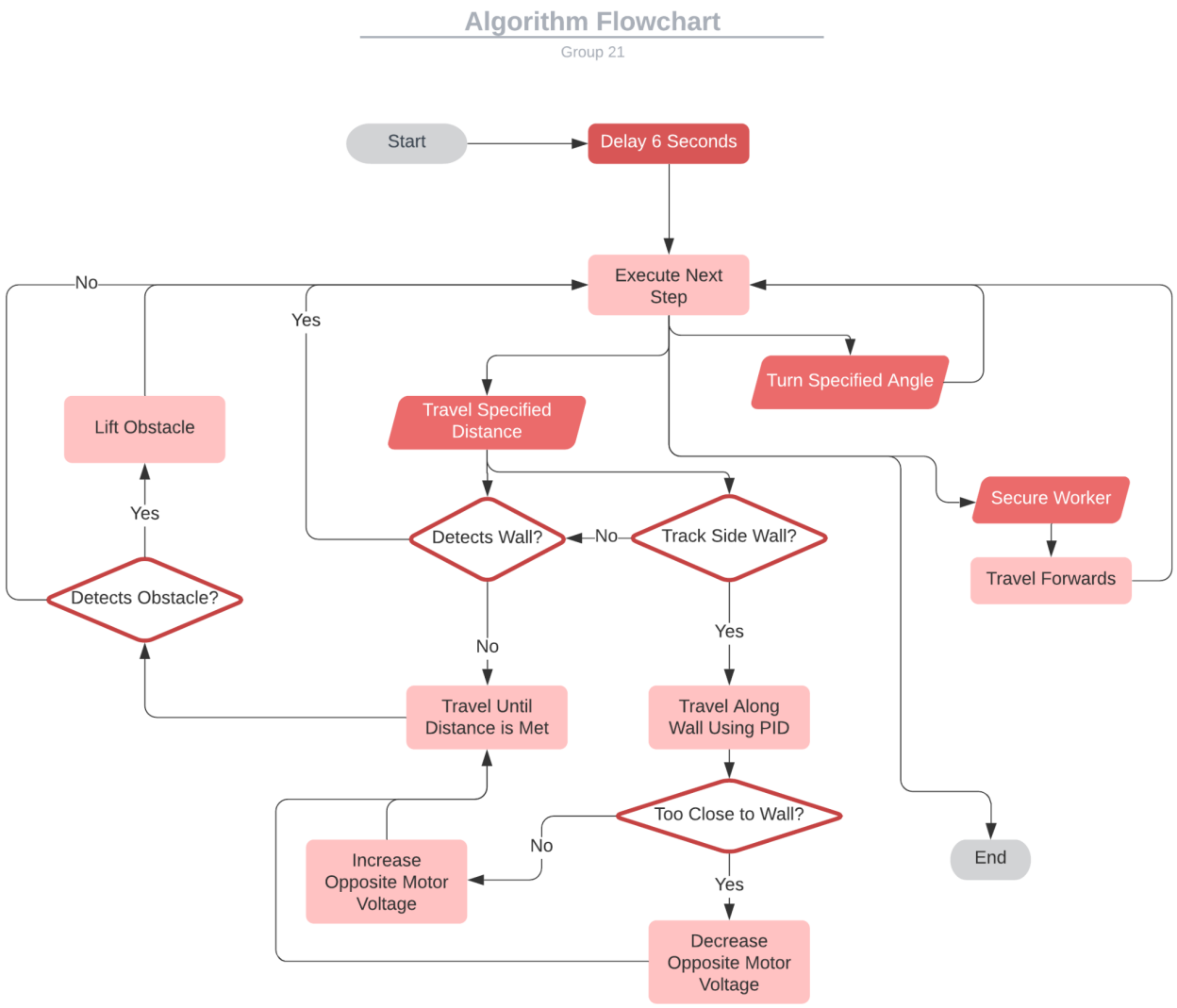


Figure 19: Algorithm Flowchart

Circuit Diagrams

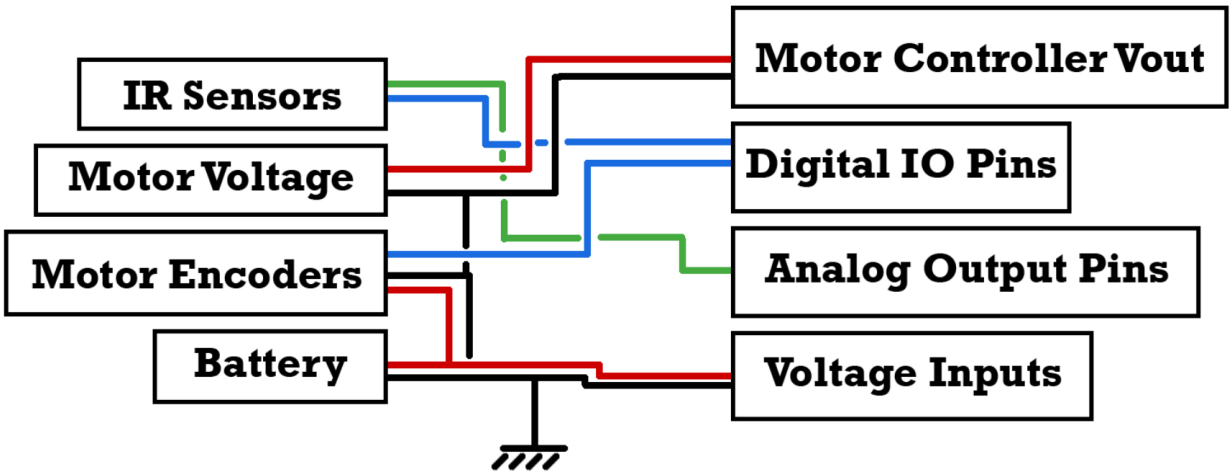


Figure 20: Circuit Diagram

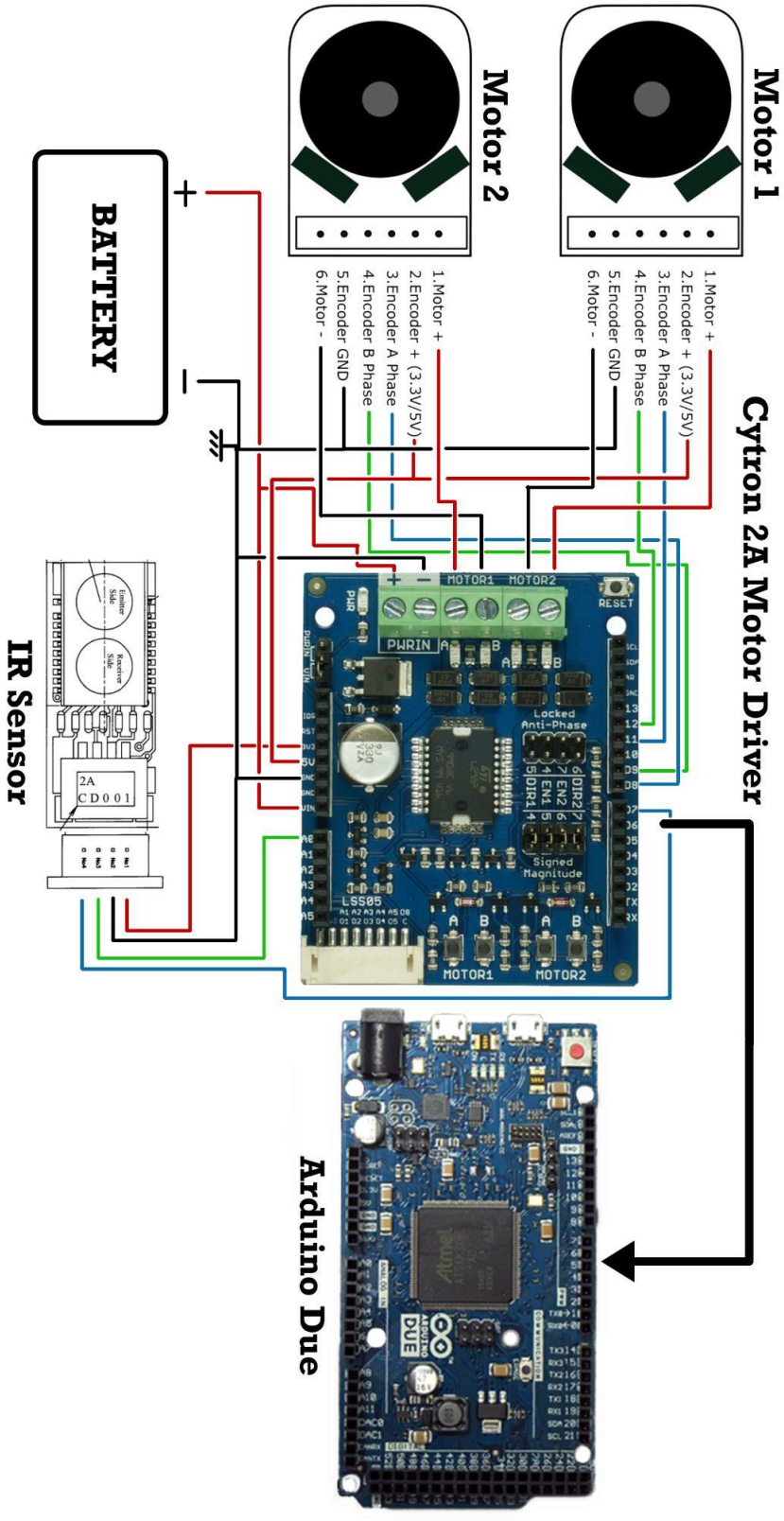


Figure 21: Detailed Circuit Diagram

Code

```
#include <AutoPID.h>
#include <Servo.h>
#include "Cytron_Shield3AMotor.h"
#include <DueTimer.h>

Shield3AMotor motor(SIGNED_MAGNITUDE);
// obstacle servo motor parameters
Servo obstacleServo;
#define OBSTACLE_SERVO_LOW 90
#define OBSTACLE_SERVO_HIGH 180
// grabber servo motor parameters
Servo gripperServo;
#define GRIPPER_SERVO_CLOSE 90
#define GRIPPER_SERVO_OPEN 200
// DC motor encoder pins
#define ENCODER_OUTPUT_1A 52
#define ENCODER_OUTPUT_1B 9
#define ENCODER_OUTPUT_2A 11
#define ENCODER_OUTPUT_2B 12

// physical constants
#define CELL_SIZE 20.0
#define CART_WIDTH 11.7
#define FRONTWALL_CHECK_VALUE 570
// IR pins
#define IR_FRONT_DIN 37
#define IR_FRONT_AO A0
#define IR_LEFT_DIN 36
#define IR_LEFT_AO A1
#define IR_RIGHT_DIN 35
#define IR_RIGHT_AO A2
#define IR_BOTTOM_DIN 28
#define IR_BOTTOM_AO A3

// PID parameters
#define OUTPUT_MAX 30
#define KP 0.96
#define KI 0.02
#define KD 0.3
double setpoint = 200, outputVal;

// IR moving average distance
#define IR_AVG_LEN 3

// true if obstacle detected
bool obstacle = false;
// turning direction to determine encoder to read
String dir = "";
// pulse count for both encoders
```

```
int pulsesCount1 = 0;
int pulsesCount2 = 0;

// turning state (left right or forward)
int turning = 0;

// IR readings arrays
double ir_reading[] = {0, 0, 0, 0};
double ir_balances[IR_AVG_LEN];
double ir_balance;
int ir_balance_index = 0;

AutoPID centerPID(
    &ir_balance,
    &setpoint,
    &outputVal,
    -OUTPUT_MAX,
    OUTPUT_MAX,
    KP, KI, KD
);

// function to initialize IR sensors
void enableIRSensors() {
  pinMode(IR_FRONT_DIN, OUTPUT);
  pinMode(IR_FRONT_AO, INPUT_PULLUP);
  digitalWrite(IR_FRONT_DIN, HIGH);

  pinMode(IR_LEFT_DIN, OUTPUT);
  pinMode(IR_LEFT_AO, INPUT_PULLUP);
  digitalWrite(IR_LEFT_DIN, HIGH);

  pinMode(IR_RIGHT_DIN, OUTPUT);
  pinMode(IR_RIGHT_AO, INPUT_PULLUP);
  digitalWrite(IR_RIGHT_DIN, HIGH);
}

// function to initialize DC motors
void initMotors() {
  pinMode(ENCODER_OUTPUT_1A, INPUT);
  pinMode(ENCODER_OUTPUT_1B, INPUT);
  pinMode(ENCODER_OUTPUT_2A, INPUT);
  pinMode(ENCODER_OUTPUT_2B, INPUT);
  attachInterrupt(
    digitalPinToInterrupt(ENCODER_OUTPUT_1A),
    readEncoder1,
    RISING
  );
  attachInterrupt(
    digitalPinToInterrupt(ENCODER_OUTPUT_2A),
    readEncoder2,
    RISING
```

```
  );
}
// encoder read functions for interrupt
void readEncoder1() {
  int b = digitalRead(ENCODER_OUTPUT_1B);
  if (b == 0) {
    pulsesCount1++;
  } else {
    pulsesCount1--;
  }
}

void readEncoder2() {
  if (turning == -1) {
    int b = digitalRead(ENCODER_OUTPUT_2B);
    if (b == 0) {
      pulsesCount2++;
    } else {
      pulsesCount2--;
    }
  }
}

// measure IR sensors and keep moving average of sensor of interest
void irSensorTracking() {
  ir_reading[0] = analogRead(IR_LEFT_AO);
  ir_reading[1] = analogRead(IR_FRONT_AO);
  ir_reading[2] = analogRead(IR_RIGHT_AO);
  ir_reading[3] = analogRead(IR_BOTTOM_AO);
  ir_balances[ir_balance_index++ % IR_AVG_LEN] = (dir == "LEFT" ? ir_reading[0] : (dir == "RIGHT" ?
ir_reading[2] : 0)) / IR_AVG_LEN;
  double sum = 0;
  for (int i = 0; i < IR_AVG_LEN; i++) {
    sum += ir_balances[i];
  }
  ir_balance = sum / IR_AVG_LEN;
  Serial.print(ir_reading[1]);
  Serial.print(" ");
  Serial.println(ir_reading[3]);
}

void setup() {
  Serial.begin(9600);
  initMotors();
  enableIRSensors();
  // attach interrupt to run every 50ms
  Timer1.attachInterrupt(irSensorTracking);
  Timer1.start(50000);
  // set PID time step to 50ms
  centerPID.setTimeStep(50);
  delay(5000); // delay to allow for placement of robot after start
```

```
}

// travel straight a distance, with string to define wall to track, safeMode
// defines the front sensor to check for walls, and obstacle sense to check
// for obstacles in path
void travelStraight(float distance, String wallTrackDir, bool safeMode, bool obstacleSense) {
  dir = wallTrackDir;
  if (wallTrackDir == "LEFT") setpoint = 205;
  else setpoint = 220;
  float distanceTravelled = 0;
  while (distanceTravelled < distance && (safeMode ? ir_reading[1] < FRONTWALL_CHECK_VALUE : true)) {
    pulsesCount1 = 0;
    // if front sensor doesnt detect object but bottom does, obstacle present
    if (ir_reading[3] > FRONTWALL_CHECK_VALUE - 20 && ir_reading[1] < FRONTWALL_CHECK_VALUE + 20 &&
obstacleSense) {
      obstacle = true;
      motor.control(0, 0);
      break;
    }
    // depending on wallTrackDir, track left or right wall (or neither)
    if (wallTrackDir == "RIGHT") {
      centerPID.run();
      // use PID output to control motor voltages
      motor.control(40 + outputVal / abs(outputVal) * 4 * sqrt(abs(outputVal)), 40 - outputVal /
abs(outputVal) * 4 * sqrt(abs(outputVal)));
    } else if (wallTrackDir == "LEFT") {
      centerPID.run();
      motor.control(40 - outputVal / abs(outputVal) * 4 * sqrt(abs(outputVal)), 40 + outputVal /
abs(outputVal) * 4 * sqrt(abs(outputVal)));
    } else {
      motor.control(47, 45);
    }
    delay(50); // only check distance every 50ms
    // calculation for pulses to distance
    distanceTravelled += 4 * PI * 3 * abs(pulsesCount1) / 2940.0;
  }
  // if obstacle detected execute lift procedure
  if (obstacle) liftObstacle();
  motor.control(0, 0);
  // stop PID calculations once no longer moving
  centerPID.stop();
  delay(500);
}

// function to execute a turn angle, highPower to overcome bumps in the path
void turn(float rad, bool highPower) {
  rad *= -0.96;
  if (rad > 0) {
    turning = -1;
    motor.control(0, highPower ? 100 : 50);
  } else {
```

```
    turning = 1;
    motor.control(highPower ? 100 : 50, 0);
  }
  // calculate arc length to travel for encoders
  float distanceToTravel = abs(rad) * CART_WIDTH;
  float distanceTravelled = 0;
  while (distanceTravelled < distanceToTravel) {
    pulsesCount1 = 0;
    pulsesCount2 = 0;
    delay(50);
    // measure distance travelled so far in arc length
    distanceTravelled += 4 * PI * 3 * abs(rad > 0 ? pulsesCount2 : pulsesCount1) / 2940.0;
  }
  motor.control(0, 0);
  turning = 0;
  delay(500);
}

// function to lower the obstacle gripper servo
void lowerObstacleServo() {
  for (int i = OBSTACLE_SERVO_HIGH; i >= OBSTACLE_SERVO_LOW; i--) {
    obstacleServo.write(i);
    delay(20);
  }
}

// function to raise the obstacle gripper servo
void raiseObstacleServo() {
  for (int i = OBSTACLE_SERVO_LOW; i <= OBSTACLE_SERVO_HIGH; i++) {
    obstacleServo.write(i);
    delay(20);
  }
}

// function to execute operation and lift the obstacle
void liftObstacle() {
  setObstacleServo(true);
  lowerObstacleServo();
  delay(1000);
  travelStraight(CELL_SIZE / 2, "", false, false);
  raiseObstacleServo();
  delay(1000);
  setObstacleServo(false);
}

// open the gripper
void gripperOpen() {
  setGripper(true);
  for (int i = GRIPPER_SERVO_CLOSE; i <= GRIPPER_SERVO_OPEN; i++) {
    gripperServo.write(i);
    delay(50);
```

```
  }
  setGripper(false);
}

// close the gripper
void gripperClose() {
  setGripper(true);
  for (int i = GRIPPER_SERVO_OPEN; i >= GRIPPER_SERVO_CLOSE; i--) {
    gripperServo.write(i);
    delay(50);
  }
  setGripper(false);
}

// attach or detach servo to pin to avoid unwanted motion
void setGripper(bool on) {
  if (on) gripperServo.attach(31);
  else gripperServo.detach();
}

// attach or detach servo to pin to avoid unwanted motion
void setObstacleServo(bool on) {
  if (on) obstacleServo.attach(33);
  else obstacleServo.detach();
}

void loop() {
  // maze layout instructions
  travelStraight(2 * CELL_SIZE, "LEFT", true, false);
  turn(PI / 2, false);
  travelStraight(6 * CELL_SIZE, "LEFT", true, !obstacle);
  travelStraight(1 * CELL_SIZE, "LEFT", true, false);
  turn(PI / 2, false);
  travelStraight(10 * CELL_SIZE, "LEFT", true, false);
  turn(PI / 2, false);
  travelStraight(10 * CELL_SIZE, "LEFT", true, false);
  turn(PI / 2, false);
  travelStraight(3 * CELL_SIZE - CART_WIDTH, "LEFT", false, false);
  turn(PI / 2, false);
  travelStraight(1 * CELL_SIZE, "", true, false);
  turn(PI / 2, false);
  travelStraight(2 * CELL_SIZE, "RIGHT", true, false);
  turn(-PI / 2, false);
  travelStraight(4 * CELL_SIZE - CART_WIDTH, "RIGHT", false, false);
  turn(-PI / 2, false);
  gripperOpen();
  travelStraight(0.5 * CELL_SIZE - CART_WIDTH, "", false, false);
  gripperClose();
  delay(999999999);
}
```

Test Plan and Results | Product Validation

The test plan for Phase A revolves heavily around lab time, but is not limited to just the time in the lab. Lower level testing is to be done during the course scheduled lab time. This time will be dedicated to confirming the function of the sensors and motors as independent systems given that the amount of time in the lab each week is limited to 2 hours. The main goal for these tests will be to convert IR sensor readings and Encoder readings into usable distance values.

Once the robot is complete, a replica of the maze with approximate dimensions will be constructed for mid level testing at home. Doing this will allow for easy testing without other groups in the way and to be able to test for as long as needed in one period of time. The main goal of these tests will be for prolonged testing of the control algorithm, and making sure that the robot is capable of basic movement, like going in a straight line and turning. These tests should prove the robots ability for navigation and ability for the sensor to run concurrently. Using approximate dimensions taken from the actual maze in the lab, a layout was created to test as many different forms of movement for the robot as possible. The layout uses multiple right and left turns, and features multiple different lengths of straight pathways. Using this, one can attempt to debug the program and ensure that the robot can move straight and make perfect 90 degree turns. A diagram of the replica maze layout can be seen below:

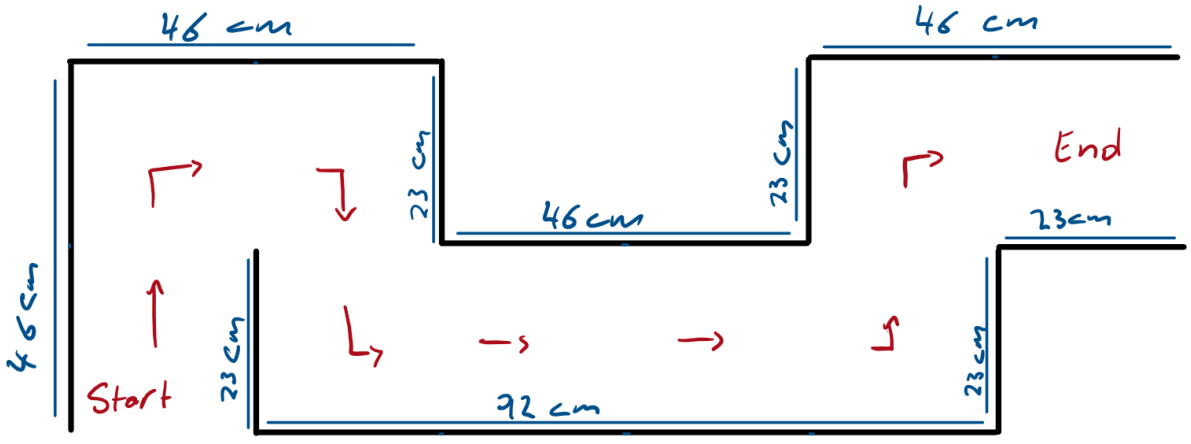


Figure 22: Replica Maze Layout

High level testing will be done in the design studio. These tests are to confirm the function of the entire system in the environment they are designed for. Once the layout of the actual maze is given, the maze in the design lab can be arranged to resemble the layout for the demo so the entire robot system can be tested. The design studio is limited to a single floor, meaning that the biggest area of potential error from testing will be the ramp on the actual maze. This ramp will only be accessible for 2 hours each week.

From the testing during Phase A of the project important areas of improvement were identified. The success of the control algorithm was verified during testing, however small issues during the manufacturing stage make the program slightly unreliable. The motors have enough room in them to be able to wiggle slightly in the cage. This means that every time the robot was picked up, the wheels would shift ever so slightly. This shift was not visible to the eye, but when running the program repeatedly, it could be seen in the form of drift to one side during forward movement, or under/over turning by a few degrees during the turns. If the motors can be more securely attached to the frame, then this issue can be eliminated. This can be accomplished by fine tuning the tolerances on the motor cages. The current tolerance was thought acceptable, being large enough to fit the motors in place with a little bit of wiggle room that could be filled in with another material if needed. Clearly the material used to fill in the gap to secure the motors was not good enough, and the tolerances will need to be adjusted.

Another area of improvement was the sensor. As outlined earlier, the plan was to have the sensor as an error check for getting too close to a wall before a turn. During the initial stages of testing, the equation for the sensor was tested separately of the motors and the algorithm for driving the robot. By the time that the whole system was able to be tested, there was an issue with using both the motors and the sensor at the same time. When running, the motors draw too much current from the battery, and the sensor does not function during motion of the robot. By the time this issue had been identified, it was too late to implement changes for the Phase A demo, but there are potential fixes to this problem. Using a separate battery just for the sensor should fix the issue, and allow the sensor to run concurrently with the motors now that they use a separate power source.

Another issue that occurred during the Phase A demo, is the gap in the floor between the top level of the maze and the beginning of the ramp. The rear bearing meant for support got caught in the gap, preventing the robot from being able to fully get itself onto the ramp to complete the maze. Initial testing did not see this issue, and probable cause for that has been determined. During initial testing on the maze, the motors were running at full power, and was able to clear the gap between the bottom floor and the bottom end of the ramp. Throughout later testing, the motor power was brought down slightly in order to decrease the speed, and increase the time between encoder pulses to increase the accuracy of the distance traveled as recorded by the motor encoders. This decrease in speed may have been what caused the robot to get stuck in the gap, as it could be that there was not enough power to pull itself through and not get stuck. Fixes for this would include a redesign of the rear support bearing system to allow for a wider surface area so that the robot does not get stuck in the gap for Phase B.

Phase B testing needs to address the issues that arose during Phase A. Further low level testing should be unnecessary, as we were able to confirm that both the motors and IR sensors can function independently. Mid level testing would need to be done again in the lab, specifically targeting the gap in the floor where the robot got stuck during the Phase A demo. This will require the redesign and implementation of the new rear support system to try and overcome the obstacle in the maze. Mid level testing would also be done on the system for securing the worker.

The replica maze, as well as time in the design lab, are now going to be used for extensive high level testing of the robot. One of the main failures of Phase A was the inability to have the sensors and the motors function at the same time. The replica maze and the design lab will be used to troubleshoot the sensors function with the motors, as well as testing the implementation of new software design specifically for the ramp. Immediately following the Phase A demo, testing was done of the Phase A design on the ramp to see if the robot would have been able to go down had it not been affected by the obstacle earlier. It was observed that the ramp surface was not flat, and that a new control algorithm was going to be needed specifically for getting down the ramp. This system would need to utilize both the motor speed control and the sensor readings. Need for the entire system categorized this as a high level test.

Parts of the mid level testing were successful. The redesigned support system was able to overcome the obstacle that limited the Phase A design, but required further post processing of the 3D print. The bottom surface of the support needed to be sanded down to ensure layer lines

didn't get caught. However, the method to secure the worker only worked under specific conditions, and will need to be adjusted for Phase C. Given that the method to secure the worker would be irrelevant if the robot couldn't reach the worker in the first place, and the timeline to the demo day, it was decided to continue with what we had and hope for the best, and focus more time and energy into the high level testing.

The high level tests took much longer, needing to tune a PID specifically for correcting any drift in forward motion that the robot would have going down the ramp. These tests however proved successful, and the robot was able to get to the bottom floor on the demo day, and through the entire bottom maze during high level tests performed in the lab time following the demo. Repeated testing of the entire system also created issues with the rear support design. As more people tested on the maze, the surface of the maze got scuffed up, increasing the friction between the rear support and the maze floor. For the Phase B demo, this was solved with a lubricant on the rear support, but it is likely that the rear support will need another redesign.

In Phase B, the concept of tracking both side walls and attempting to maintain a zero difference in distance between the two walls was used. Upon testing, this idea will likely be insufficient, as it does not allow for compensation for drift in the majority of sections in the maze. In its place, for Phase C, a new method from a previous concept to track only one wall at a time, and have an option to select which wall to track, will be implemented. This can reuse much of the PID code, but will involve a total redesign of the method in which the travel function, and the tracking itself, is done.

There also needs to be some effort towards linearizing the velocity of the cart with respect to the motor voltages. Currently, the final 10% of the motor voltage output corresponds to about the final 30% of the speed of the cart, where the previous 90% only accounts for a minor difference in speed. The result of this is that the PID will not make drastic enough changes to small deviations from the set point, but will turn too drastically when far from the set point. In order to do this, for Phase C a linearization technique will be used. This technique will take low values and modify them to be higher, and lower high values so that they do not create too dramatic a shift in voltage too suddenly. The two options for ways to do this are inverse exponential functions, and square root functions. Since square root functions are less computationally intensive, they are the most likely candidate for the linearization function. By

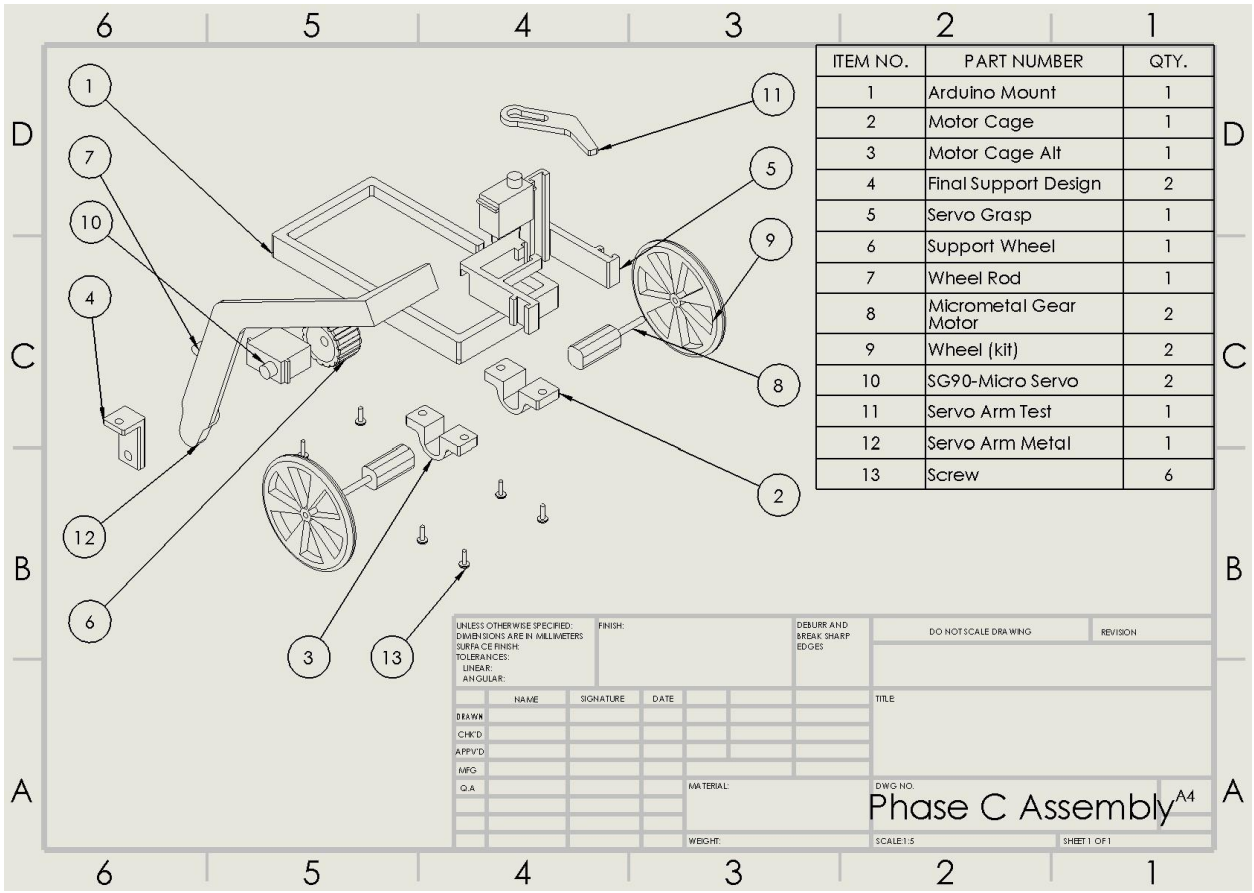
using this function, it can make the output from the PID more predictable, and improve response time.

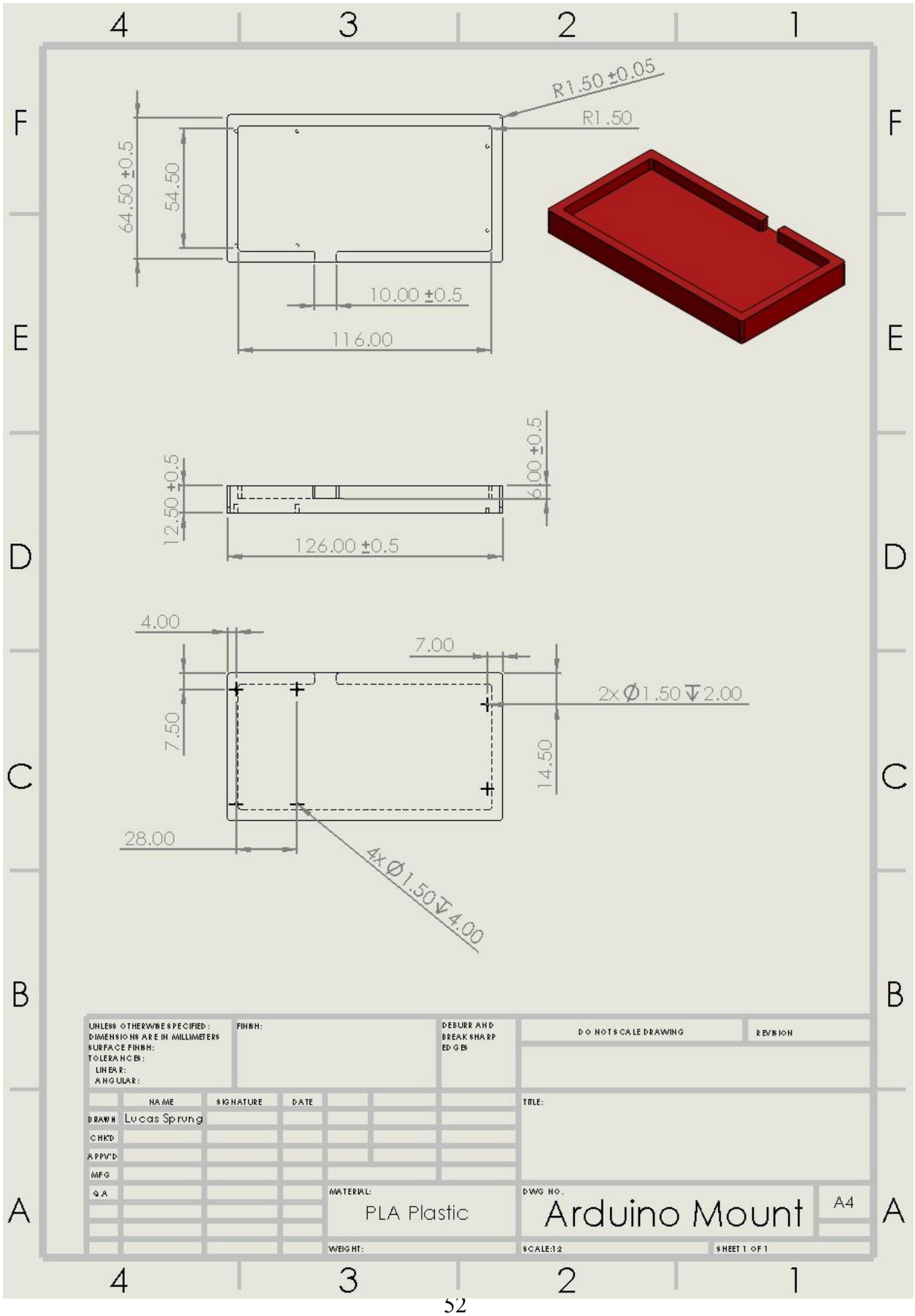
The ability to sense obstacles was tested once the fourth IR sensor was implemented in Phase C. The testing consisted of finding an obstacle of approximately the same height to the expected obstacles, and testing reliability of the obstacle sensing patterns. The robot was able to detect obstacles with a fairly high reliability, and also did not have many issues with accidentally sensing obstacles when there was none. Overall, some minor improvements could be made to the body to avoid IR sensors interfering with each other, but the results were not underperforming in this respect.

Testing of the method of dealing with obstacles required access to the design lab, as the LEGO Duplo blocks were not accessible outside of the design lab. During these testing stages it was found that the actuated arm could pick up the block, but only under very specific conditions. The results were not as repeatable as they should have been. Issues with the battery also became concerning during the high level testing for phase C. Ultimately more time was needed to refine these designs.

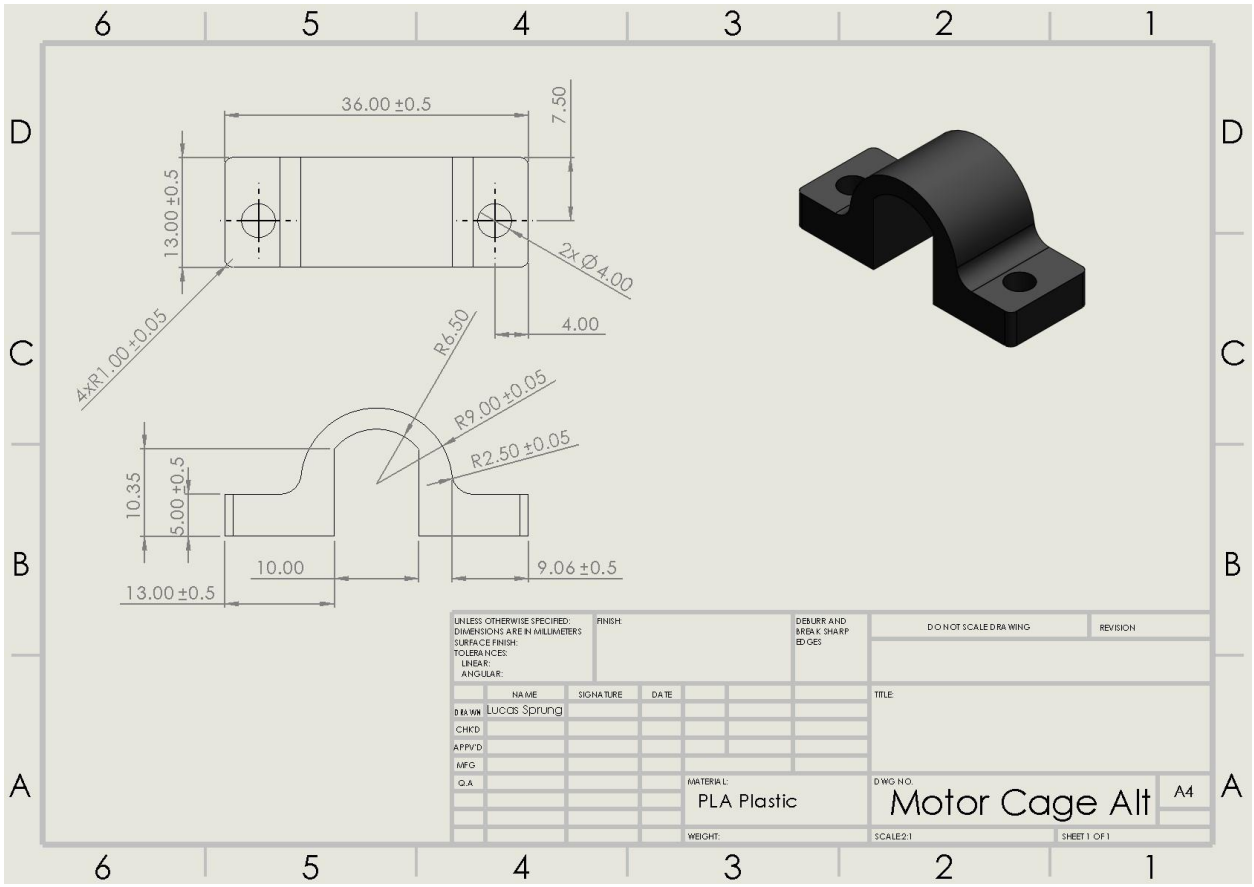
The new servo actuated method of securing the worker was tested both in and out of the design lab, and was able to reliably secure the worker with good repeatability.

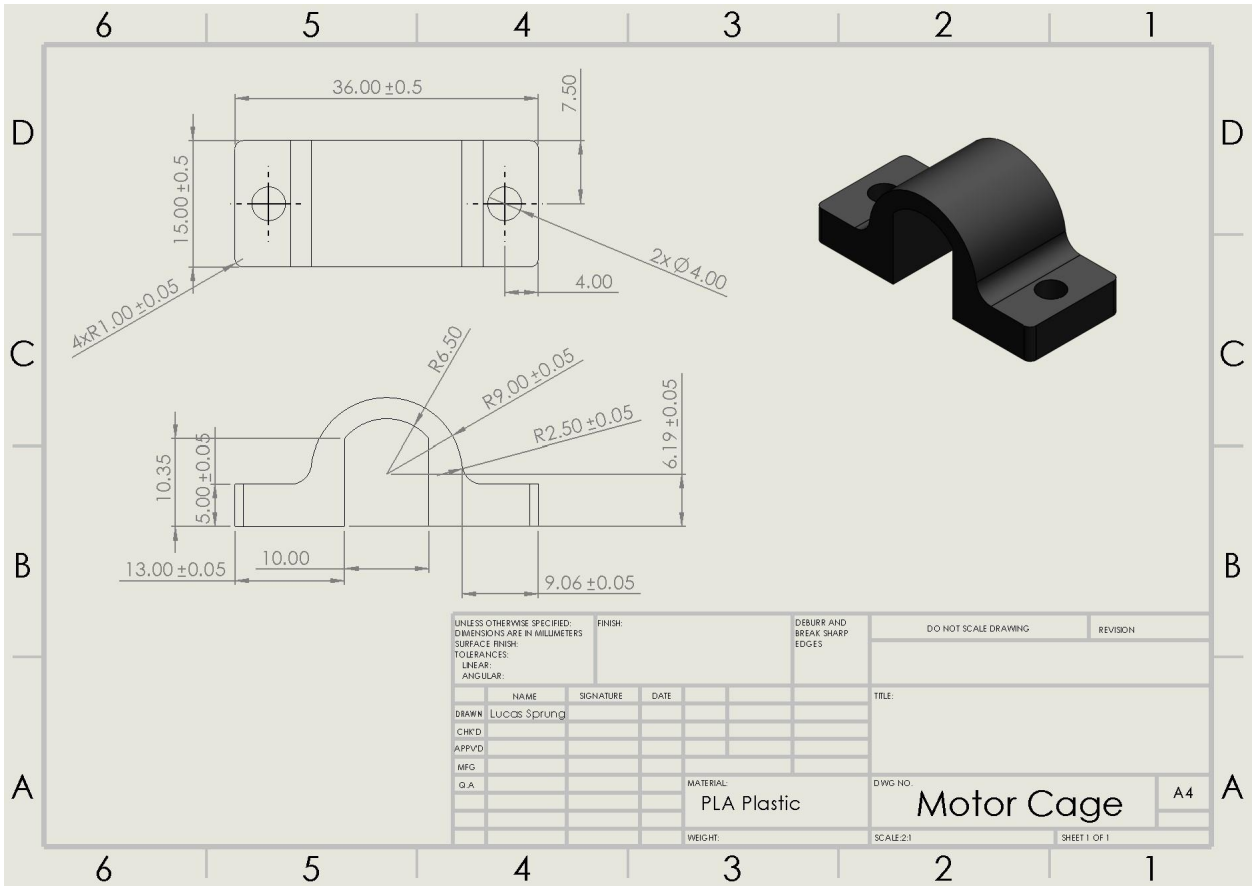
Part Drawings

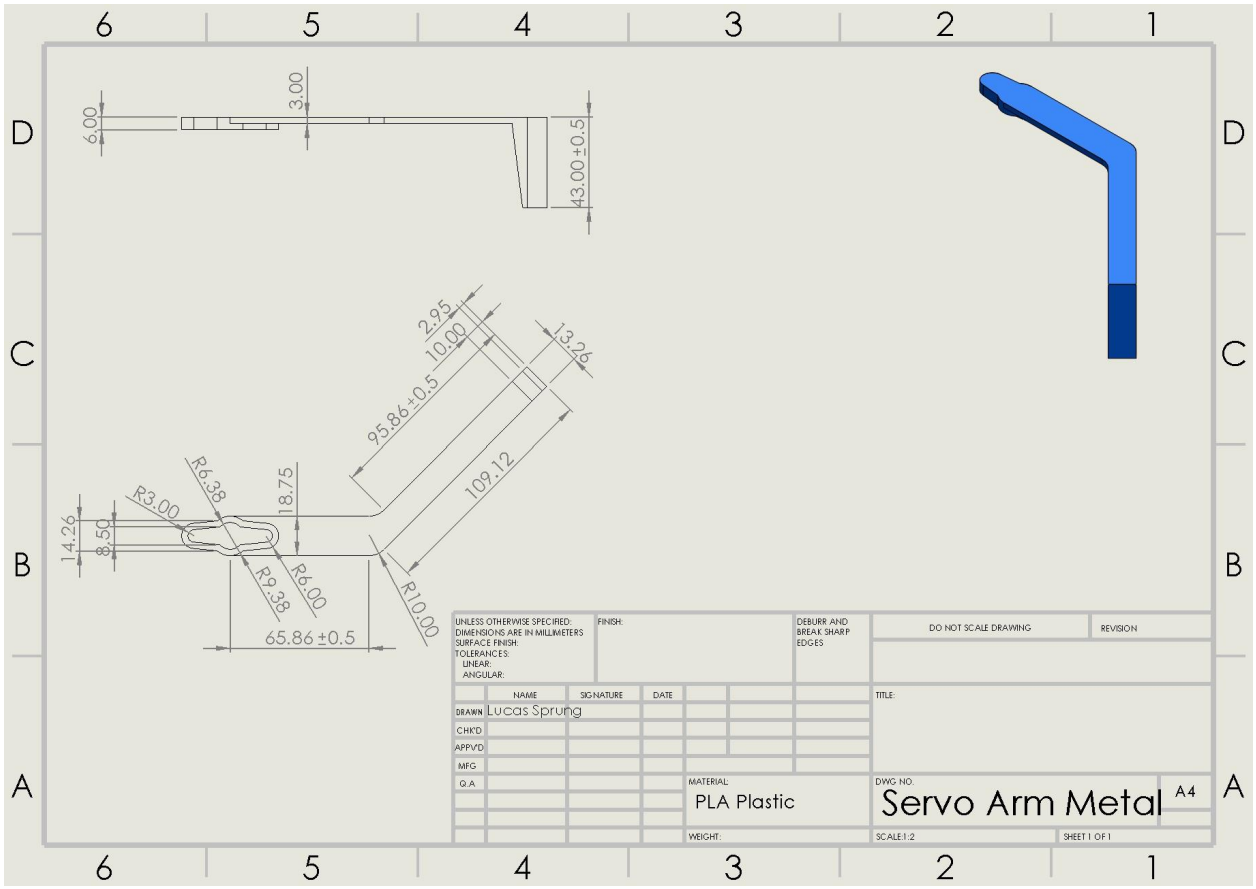


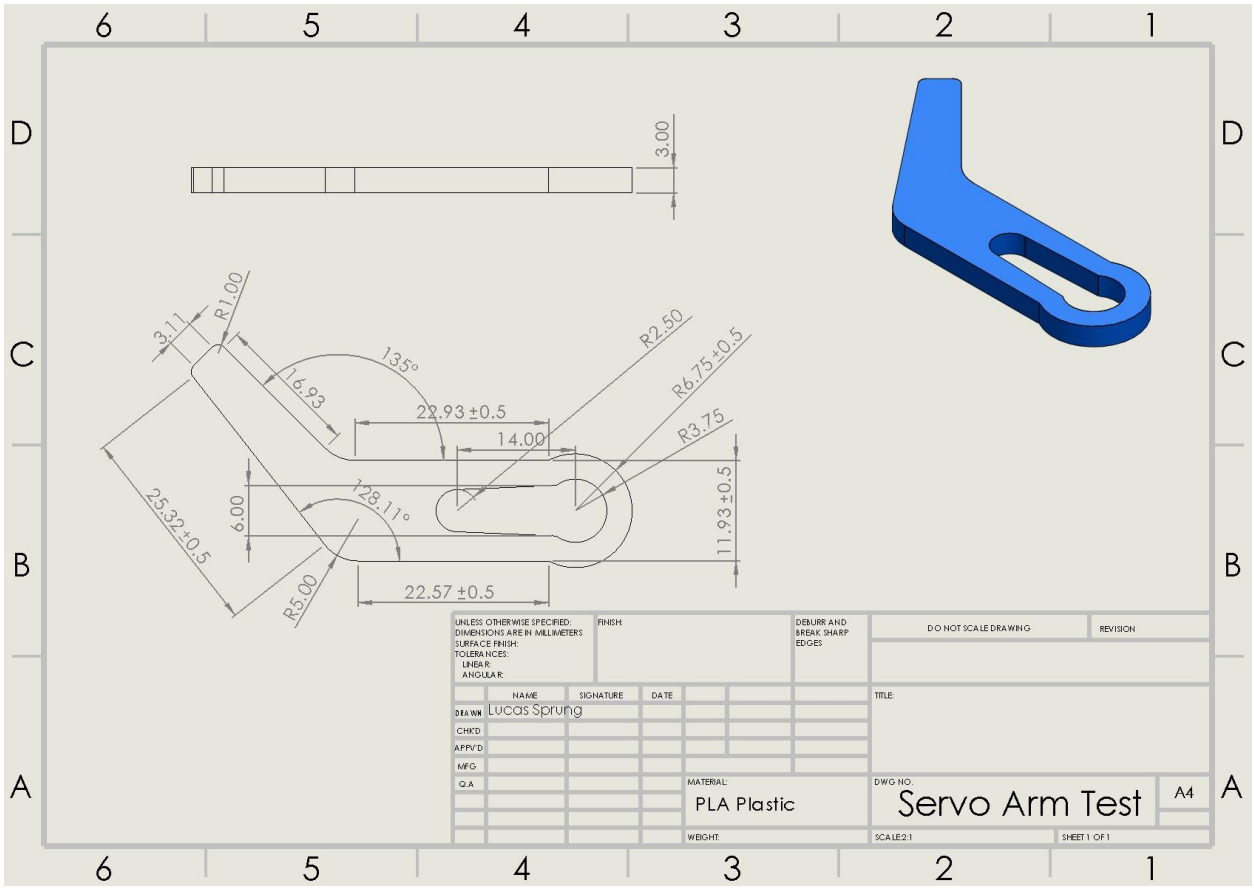


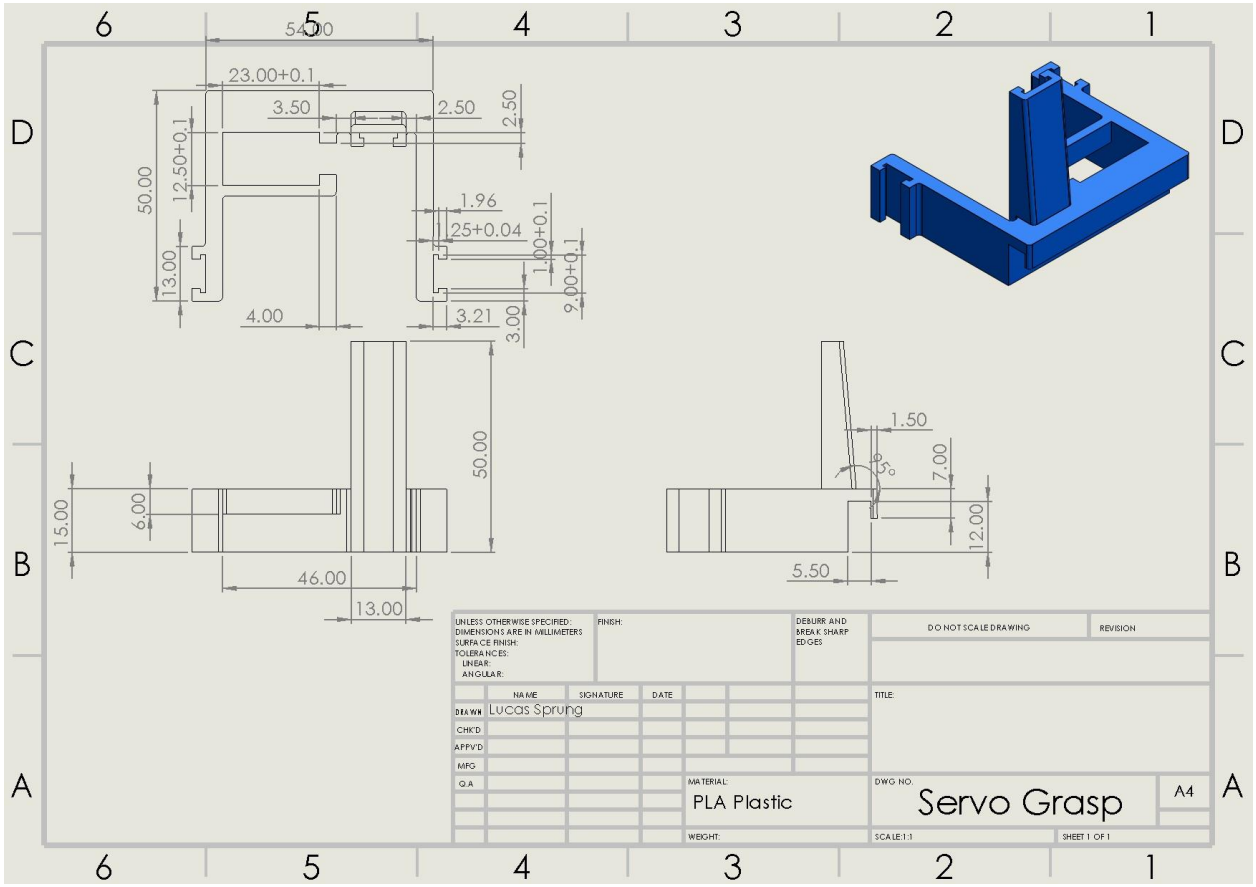
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION	
	NAME	SIGNATURE	DATE	TITLE:	
DRAWN	Lucas Sprung				
CHKD					
APPV'D					
MFG					
Q.A					
		MATERIAL: PLA Plastic		DWG NO. Arduino Mount	A4
		WEIGHT:		SCALE:1:2	SHEET 1 OF 1

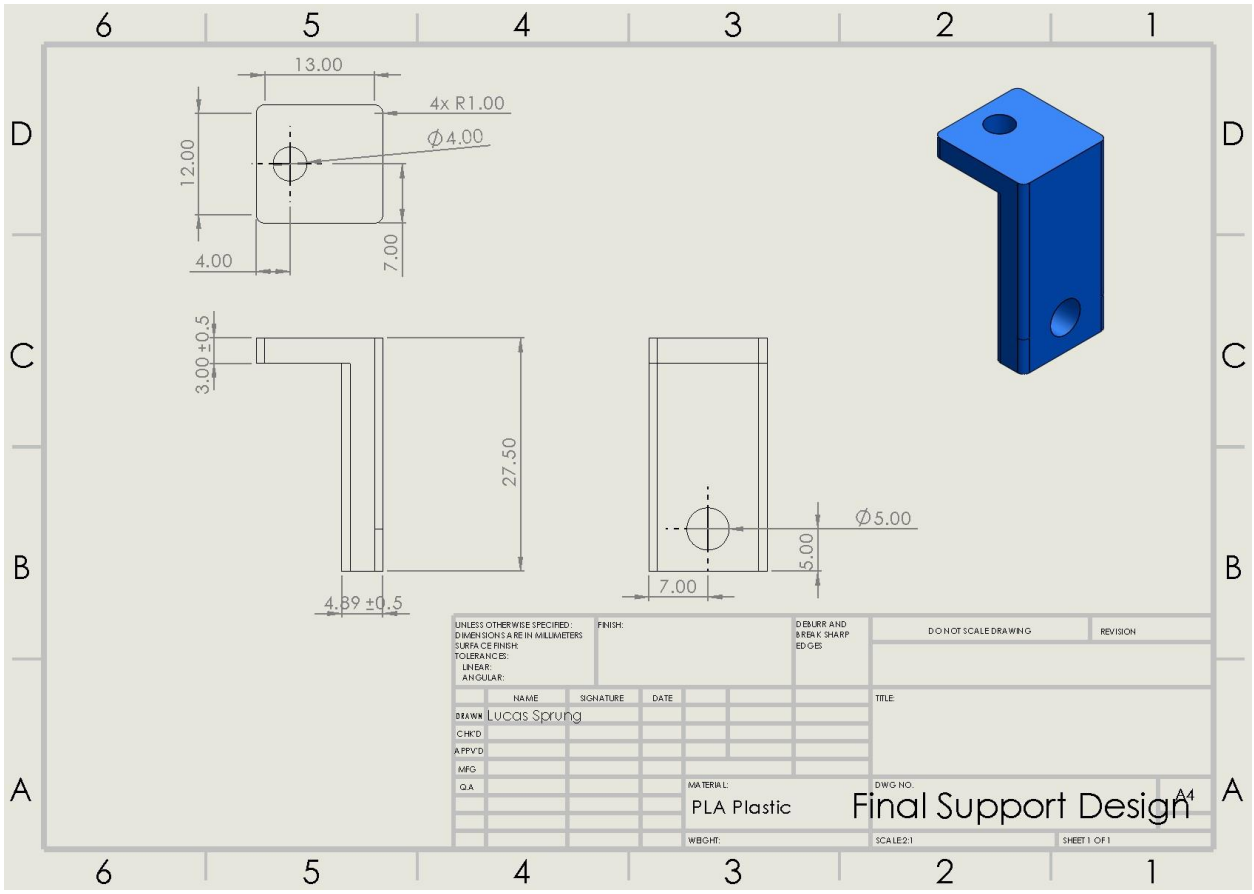

36.00 ±0.5
7.50
13.00 ±0.5
2x Ø4.00
4.00
4XR1.00 ±0.05
R6.50
R9.00 ±0.05
R2.50 ±0.05
10.35
5.00 ±0.5
10.00
9.06 ±0.5
13.00 ±0.5
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
Lucas Sprung
CHKD
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
PLA Plastic
DWG NO.
Motor Cage Alt
A4
WEIGHT:
SCALE:2:1
SHEET 1 OF 1

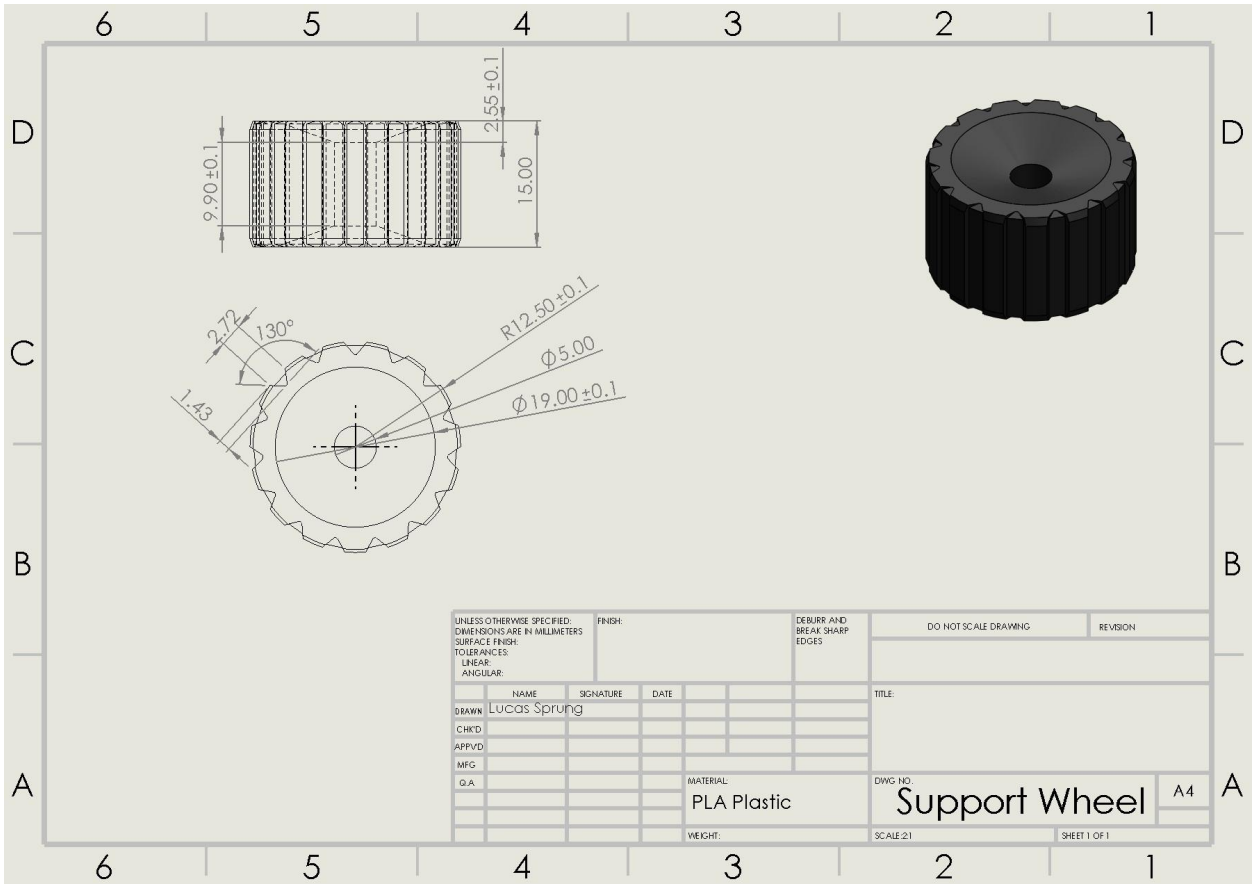

36.00 ±0.5
7.50
15.00 ±0.5
2x Ø4.00
4.00
4xR1.00 ±0.05
R6.50
R9.00 ±0.05
R2.50 ±0.05
6.19 ±0.05
10.35
5.00 ±0.05
13.00 ±0.05
10.00
9.06 ±0.05
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
Lucas Sprung
CHK'D
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
PLA Plastic
DWG NO.
Motor Cage
A4
WEIGHT:
SCALE:2:1
SHEET 1 OF 1


6.00
3.00
43.00 ±0.5
2.95
10.00
13.26
95.86 ±0.5
109.12
R6.38
R3.00
18.75
14.26
8.50
R9.38
R6.00
R10.00
65.86 ±0.5
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
Lucas Sprung
CHK'D
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
PLA Plastic
DWG NO.
Servo Arm Metal
A4
WEIGHT:
SCALE:1:2
SHEET 1 OF 1

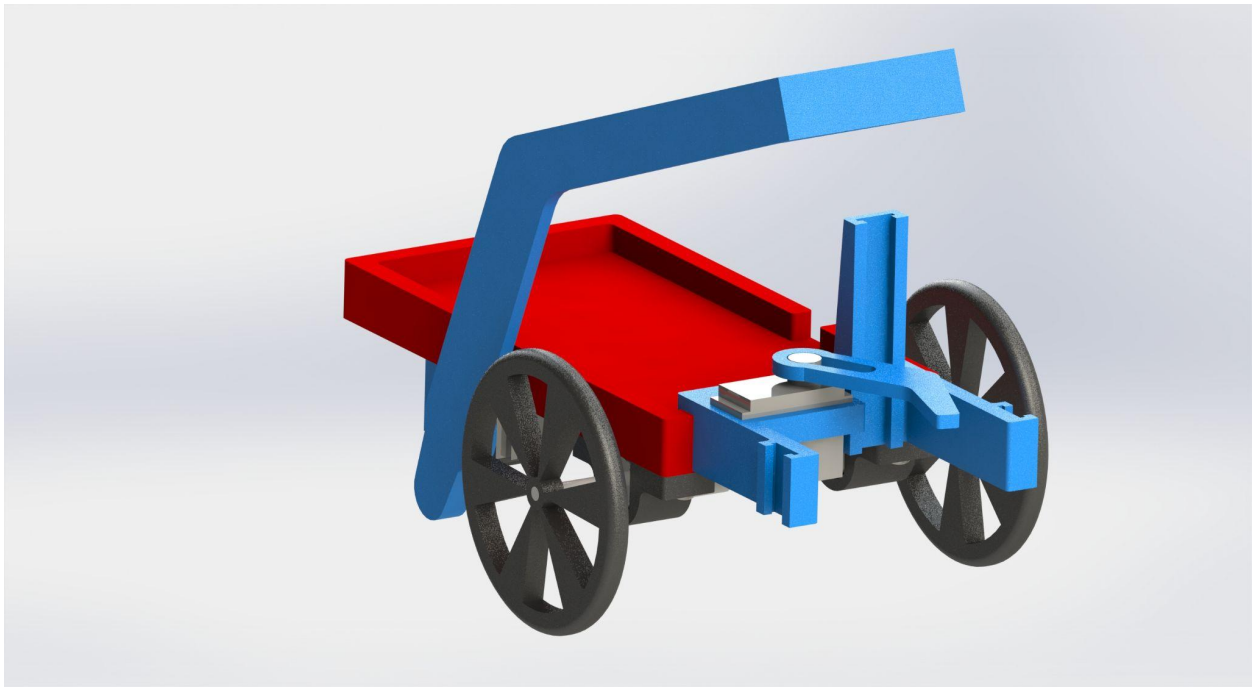

3.00
3.11
R1.00
16.93
135°
22.93 ±0.5
14.00
R2.50
R6.75 ±0.5
R3.75
11.93 ±0.5
128.11°
6.00
25.32 ±0.5
R5.00
22.57 ±0.5
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
Lucas Sprung
CHK'D
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
PLA Plastic
DWG NO.
Servo Arm Test
A4
WEIGHT:
SCALE:2:1
SHEET 1 OF 1

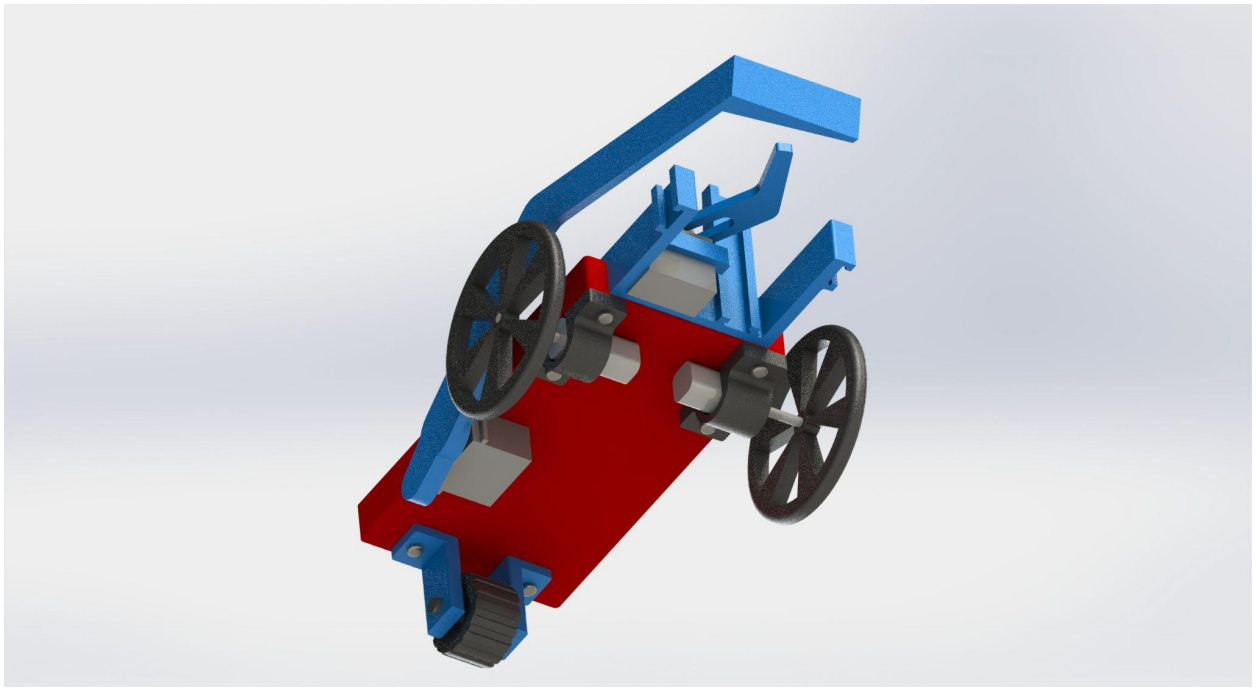

54.00
23.00+0.1
3.50
2.50
2.50
50.00
12.50+0.1
13.00
1.96
25+0.04
1.00+0.1
9.00+0.1
4.00
3.21
3.00
50.00
15.00
6.00
46.00
13.00
1.50
6
3°
7.00
12.00
5.50
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
Lucas Sprung
CHK'D
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
PLA Plastic
DWG NO.
Servo Grasp
A4
WEIGHT:
SCALE:1:1
SHEET 1 OF 1


13.00
4x R1.00
Ø4.00
12.00
4.00
7.00
3.00 ±0.5
27.50
4.89 ±0.5
Ø5.00
5.00
7.00
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
TITLE:
DRAWN
Lucas Sprung
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
PLA Plastic
DWG NO.
Final Support Design
A4
WEIGHT:
SCALE:2:1
SHEET 1 OF 1


2.55±0.1
9.90±0.1
15.00
2.72
130°
1.43
R12.50±0.1
Ø5.00
Ø19.00±0.1
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
Lucas Sprung
CHK'D
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
PLA Plastic
DWG NO.
Support Wheel
A4
WEIGHT:
SCALE:2:1
SHEET 1 OF 1

Renders





References

[1] 2017. Micro Metal Geared motor w/Encoder - ^v 75RPM 210:1. [ebook] DFROBOT, p.1-2. Available:https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/FIT0485_Web.pdf [Accessed 12 October 2021].

[2] n.d. *SHARP GP2Y0E02A*. [ebook] p.1-9. Available: https://jp.sharp/products/device/doc/opto/gp2y0e02a_e.pdf [Accessed 12 October 2021].

[3] R. Mitchell, "A comparison of popular Arduino boards: Arduino," *Maker Pro*, 12-Oct-2021. [Online]. Available: https://maker.pro/arduino/tutorial/a-comparison-of-popular-arduino-boards. [Accessed: 12-Oct-2021].

[4] S. J. Bigelow, "What is ARM processor? - definition from whatis.com," *WhatIs.com*, 16-Jan-2015. [Online]. Available: https://whatis.techtarget.com/definition/ARM-processor#:~:text=An%20ARM%20processor%20is%20one,Advanced%20RISC%20Machines%20(ARM).&text=The%20ARM%20processor's%20smaller%20size,suitable%20for%20increasingly%20miniaturized%20devices. [Accessed: 12-Oct-2021].

[5] 2015. *Cytron 2A Motor Driver Shield*. [ebook] p.1-15. Available: https://www.technobotsonline.com/Datasheets3/1519-002-User.pdf [Accessed 12 October 2021].

[6] "Lithium-Ion Battery," *Clean Energy Institute*, 25-Sep-2020. [Online]. Available: https://www.cei.washington.edu/education/science-of-solar/battery-technology/. [Accessed: 12-Oct-2021].

[7] "Standard gripper kit B - straight Mount," *ROB-13175 - SparkFun Electronics*. [Online]. Available: https://www.sparkfun.com/products/retired/13175. [Accessed: 05-Dec-2021].

[8] "Buckets - excavator," *https://www.cat.com/en_US/products/new/attachments/buckets-excavator.html*. [Online]. Available: https://www.cat.com/en_US/products/new/attachments/buckets-excavator.html. [Accessed: 05-Dec-2021].

[9] "AT340 sweeper arm - replacement," *at340*. [Online]. Available: https://www.kenwoodchefrestore.co.uk/shop/kenwood-at340-sweeper-arm. [Accessed: 05-Dec-2021].

[10] "3DOF articulated robotic arm," *HAFIZ MUHAMMAD NABEEL*. [Online]. Available: http://hmnabeel.weebly.com/3dof-articulated-robotic-arm.html. [Accessed: 05-Dec-2021].

[11] "Backtracking: Introduction," *GeeksforGeeks*, 09-Feb-2021. [Online]. Available: https://www.geeksforgeeks.org/backtracking-introduction/#:~:text=Backtracking%20is%20an%20algorithmic%2Dtechnique,reaching%20any%20level%20of%20the. [Accessed: 12-Oct-2021].

[12] "Maze solving algorithm," *Swuecho Wiki*. [Online]. Available: https://swuecho.fandom.com/wiki/Maze_solving_algorithm. [Accessed: 12-Oct-2021].

[13] "Brief communication - citeseerx.ist.psu.edu." [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.650.1709&rep=rep1&type=pdf. [Accessed: 12-Oct-2021].

[14] "Coal mine rescue robots based on Binocular Vision: A review of the state of the art," *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9141234. [Accessed: 12-Oct-2021].