Faculty of Engineering

Department of Electrical and Computer Systems Engineering

ECE3091 Engineering Design

Semester 2 2021

Group 39:

Jack Coleman 27765504 jcol38@student.monash.edu

Cameron Hancock 30606381 chan0028@student.monash.edu

Zhaoning Lu 29386993 zluu0020@student.monash.edu

Sean Martin 30586232 smar0051@student.monash.edu

Mihin Perera 29807395 mper0023@student.monash.edu

1.0 Executive Summary

This project required the development of a robot platform able to collect and store a ball bearing of a known diameter and colour within a walled arena. It was known that the ball was ferrous metallic and therefore would be attracted towards a magnetic field. It was also known that there would exist distractor bearings that the robot had to avoid and not collect.

The challenges faced in developing the robot within the context of this goal were as follows:

- Limited time to collect the target bearing
- Presence of distractor bearings
- Presence of an obstacle to avoid within the arena
- Penalties for collisions of robot with wall
- Penalties for collecting incorrect target

In order to meet these challenges, the following feature set was implemented within the developed robot platform

- Path planning optimisation within the path planning block a seen in Figure 1.1 to minimise travel time to the target
- Obstacle avoidance through detection obstacles by the ultrasonic sensors which are then fed into the path planning calculations with a key innovation being a unique obstacle avoidance strategy allowing the robot to navigate around obstacles dynamically
- Robust computer vision able to discern between target and distractor bearings performed through the target detection block in Figure 1.1
- Localisation from on-board sensors and kinematic models to accurately determine the location of the robot within the arena to prevent collisions performed within the motor block
- Accurate and reliable target bearing manipulation and collection using electro-mechanical systems, namely an electromagnet and servo in cooperation.

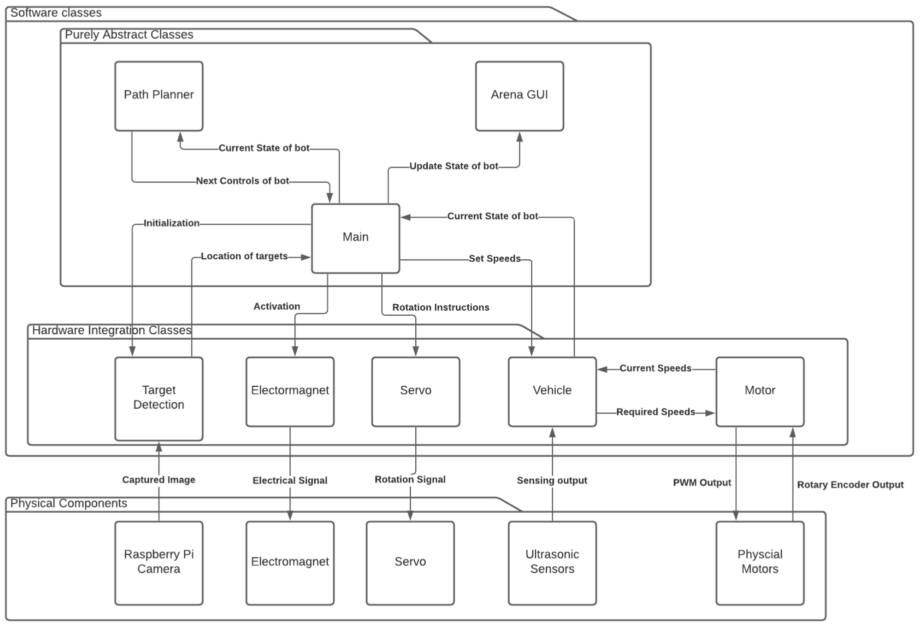

Figure 1.1 System block diagram

All of these systems were packaged into a small self-contained, robust robot able to meet the requirements of this budget with minimal cost of $329.97 without compromising on project goals. Note that this is the cost for a single prototype, and significant reductions could be had in bulk quantities.

2.0 Detailed Project Development

2.1 Mechanical Design

2.1.1 Overall Mechanical Design

The mechanical design of the robot was focused on an actuating arm to collect the targets. It used a gearbox of ratio 114.7:1 to power both the front wheels, with a bearing in the centre at the back for ease of turning. Two ultrasonic sensors were used in pair at an angle of ∓25° from the centre axis of the robot to detect walls and objects that the robot could be heading towards. The pi camera was used to detect the target using IR imaging, this was positioned high on the robot, about 19cm off the ground at an angle of 58°. Finally, the arm of the robot was operated through a servo at the base, and an electromagnet at the tip. The arm was positioned to the side of the robot to allow for minimum length, and thus, a minimum moment on the servo. Once the robot was in position, the electromagnet would be switched on, and the servo would lower the arm; the magnet would then collect the target and the servo would raise the arm back up again. These components are shown in Figures 2.1.1.1 and 2.1.1.2.

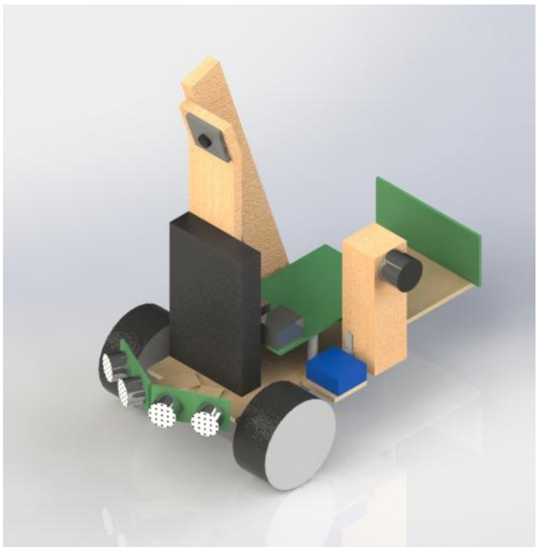
Figure 2.1.1.1 - Overall Design

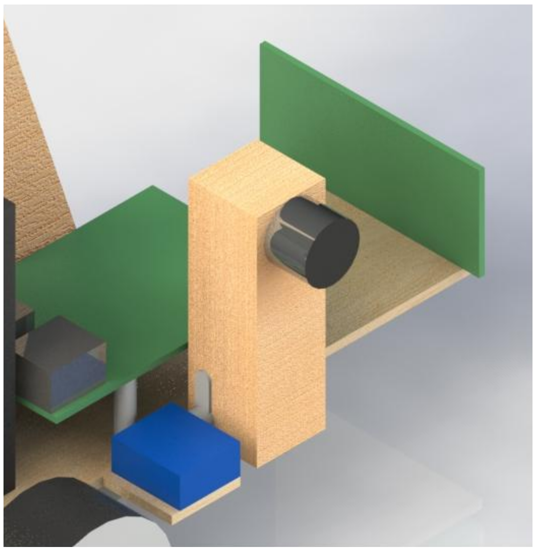
Figure 2.1.1.2 - Close up of arm design

2.1.2 Construction Materials

The materials used for this project were mainly holed perspex, veroboard and balsa wood. The holed perspex was used as a base for the robot, this allowed for ease of construction by being able to attach any component with screws. The veroboard was used for construction of external circuitry, and the balsa wood was used in affixing the camera at a certain height and angle, as well as for the arm. Balsa was the chosen wood as it is lightweight and strong under bending force, this allows for less of a moment on the servo, allowing it to operate.

2.1.3 Design Specifications

For the gears, we tested different gear ratios, and decided on a ratio of 114.7:1 as it was high enough to power the wheels at a slow rate, allowing for more precise movement.
The angle of the camera was experimented with and a final angle of 58° was chosen. This allowed for vision close to the robot while still being able to see the farthest point of the arena.
The position of the arm was decided to be on the side of the robot to minimize the length of the arm, thus minimizing the moment on the servo. Originally it was set up to lower over the front of the robot but it had to go over the ultrasonic sensors, requiring a long arm that the servo was unable to handle.
The original position of the battery pack was underneath the rear of the robot, but this made the robot too back heavy, leading to it slipping whilst driving, making the movement less accurate. To counteract this, the battery pack was moved to in front of the camera structure, giving more weight to the front of the robot, leading to more precise movement.

This design requires quite precise and accurate maneuvering as its area for picking up the target is quite small. However, due to our high gear ratio, slow movement, precise measurements and rigorous optimization, the robot is able to consistently get in the area to be able to collect the target.

2.1.4 Alternative Designs

An alternate design that we deliberated on was to attach a spinning wheel to the front of the robot, scooping up the target and filtering it so that no distractors were collected. This would require less precision on driving and target detection. However, this added extra complexity to the design, by requiring more servos and power draw. Furthermore, it was unclear how the system would handle multiple distractors or targets at the same time, possibly jamming causing the system to fail. Due to these factors, it was decided that this design would not be used as the multiple points for failure would introduce unnecessary risk to the system.

Another alternative design that was proposed was to use a pulley system to collect the target. This would involve using pulleys to move a plate with the target above the robot and drop it into a sorting system. This design would reduce the complexity of power management and make it easier to store the target, however it added a large amount of complexity to the robot and would obstruct the front of the robot which would affect the vision and detection systems. Therefore this design was ultimately rejected in favour of the chosen arm and electromagnet design.

2.2 Electronic Design

2.2.1 Overall Electronic Design

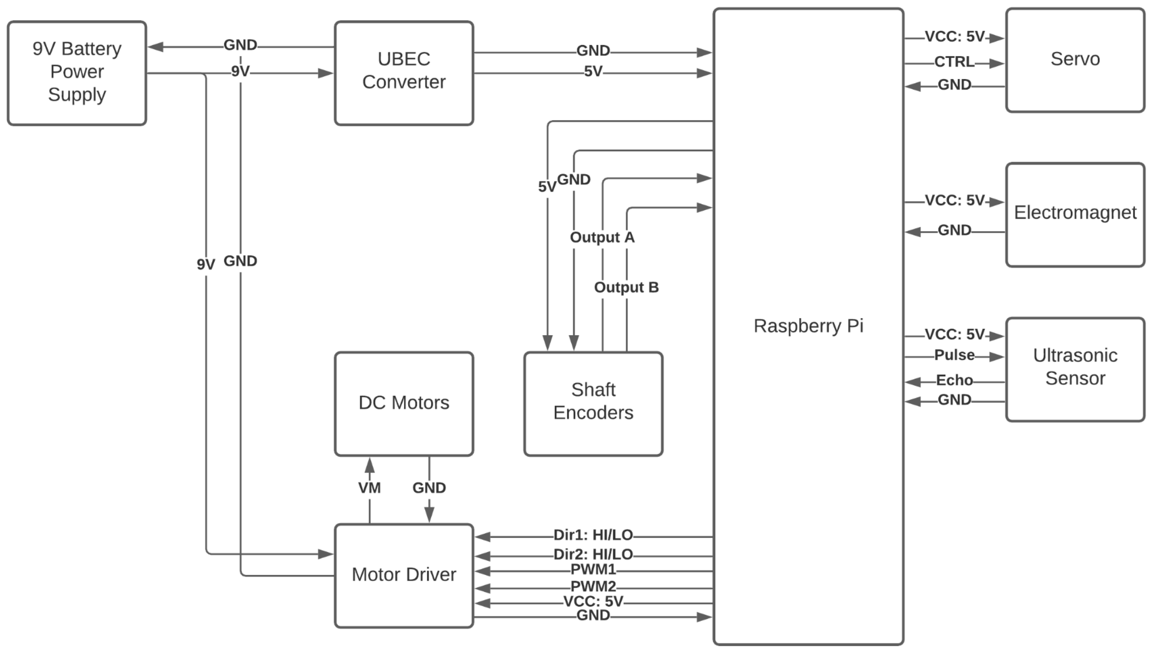

Figure 2.2.1.1 System diagram for electronic design

The key design principle behind the electronics was the principle of modularity. Due to the limitations of COVID it was necessary to test and develop the electronics systems in isolation before they were integrated into the robot, and this was accomplished by leveraging the GPIO header pin array on the Pi. This enabled most of the components of the electronics system to be wired point-to-point, with minimal soldering required. Figure 2.2.1.1 shows the broad system design, with the relevant connections, demonstrating the point-to-point connections between the ultrasonic sensors and small actuators (electromagnet and servo), as well as the IO signals sent to the propulsion subsystem consisting of the shaft encoders, DC motors and drivers.

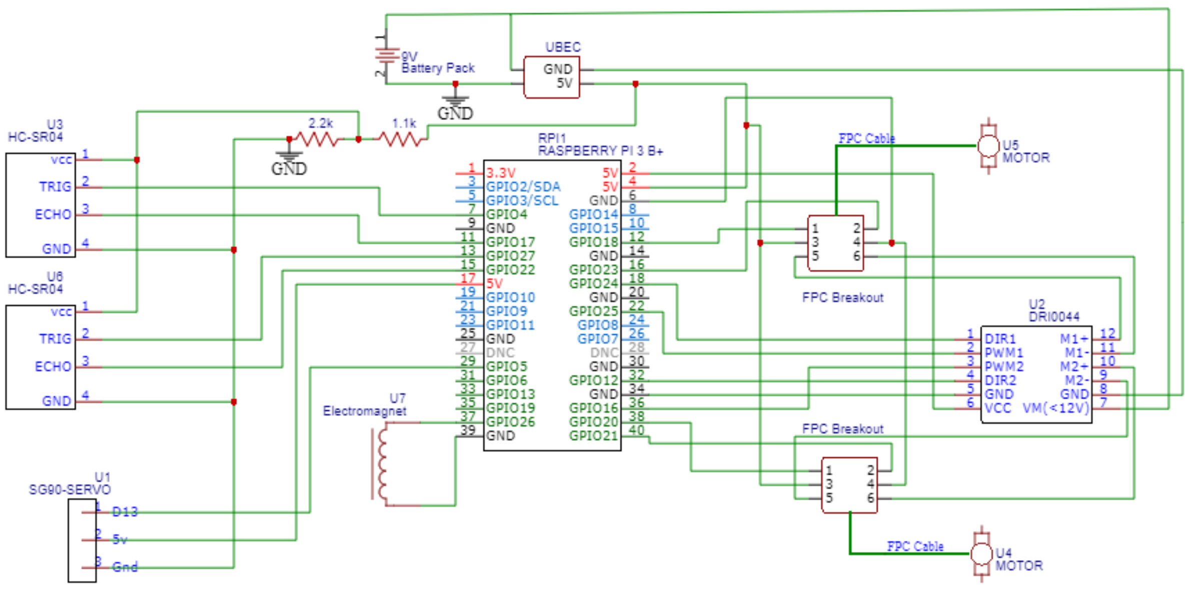

Figure 2.2.1.2 Schematic of electronic system

2.2.2 Actuator and Sensor Design

The design of the systems surrounding the actuators and sensors were done concurrently with requirements from the mechanical and software design. This allowed the use of an agile development strategy which resulted in multiple iterations of the electronic system to meet the design requirements. The first requirement was the robot's knowledge of its own position in space as a function of its own movement. The odometry calculations hence required the measurement of the speed of the DC motors which resulted in the implementation of shaft encoders. These shaft encoders were interfaced with using an FPC breakout board via a motor driver to the Pi header pins, and the implementation of the shaft encoders within the drivetrain will be discussed in Section 2.2.3.

The next sensors implemented were the ultrasonic sensors, as discussed previously these were added in order to meet the requirement of the robot being able to detect and then avoid (through software) obstacles such as walls and (in the original design brief) another robot in the arena. The sensors required the supply voltage to be stepped down from 5V to 3.3V via a two-resistor voltage divider which was soldered on veroboard to prevent header cables and components disconnecting. The mechanical design for the grabber (as mentioned in Section 2.1) required the use of an electromagnet and servo; these proved to be simple to implement in the robot and were wired directly to the Raspberry Pi via the GPIO header cables.

An alternative proposed for the sensors was the use of a LIDAR, this would be much more accurate and be able to detect smaller objects at shallower angles. However this was not ultimately implemented as LIDAR's are very expensive, power consuming complex systems which are beyond the scope of this project. Another proposition was to use the camera to detect walls, however this would have required a more complex computer vision system in the software stack and would also have required multiple cameras to cover all horizontal axes of movement. The use of the ultrasonic sensors proved to be the most cost-effective and easiest to implement of the sensing systems and was therefore used in the final product An alternative to the actuator for grabbing the bearing was a claw system comprised of multiple servos, however this would have required control of multiple servos which would have been complex to implement and an over engineered solution to a relatively simple problem. The use of an electromagnet allowed the robot to differentiate between bearing and distractors even in the final stage grabbing the target, which increased the robustness of the grabbing system all the way down to a hardware level.

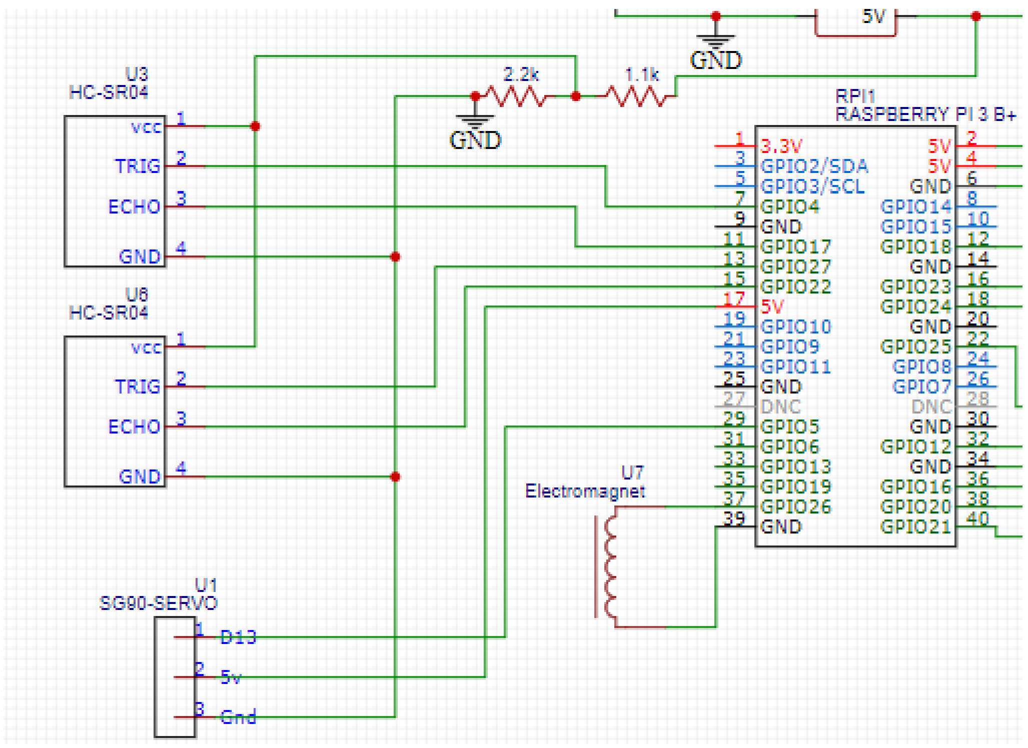

Figure 2.2.2.1 Schematic of sensors and actuators

2.2.3 Propulsion Design

The propulsion/drivetrain proved to be the most complex part of the electronic design. The system consisted of a dual motor differential drive with a castor wheel allowing for a tight turning circle and simple motor control. The software stack outputted signals that determined the direction of the motor as well as the speed via PWM to the motor driver. This motor driver acted as an H-bridge, providing logic that supported changing direction of the DC motors through logical signals sent by the Pi which enabled the motors to be powered by the 9V battery supply and yet still be fully controlled by the Pi without dedicating power pins. The motor driver then sent information directly to the motors via the FPC 6-pin breakout boards. These boards were necessary as by implementing the rotary shaft encoders the motors and encoders could only be interfaced via an FPC flat cable connector. The important innovation in this design was allowing the motors to be powered by the batteries at

9V whilst still powering the boards and encoders with the 5V supply and not oversaturating the power input. In order to increase the robustness of this relatively complex system the motor driver and FPC pinout boards were soldered onto veroboard and then affixed onto the chassis to prevent header pins from disconnecting randomly. The PWM output to the motors had to be capped at approximately 50%, as the encoders began to skip steps and not read the speed of the motors correctly as the speed increased.

An alternative design considered for the propulsion system was the use of a 4-wheel design system. This would have allowed for a more stable platform with differential drives for each axle. Whilst allowing for more control of the robot this would have increased the complexity of the drivetrain system, as differential control would have been required for 4 separate motors, and implementing servos as a steering mechanism would also have made the system more complex. Most importantly, within the context of the competition, a 4-wheel differential drive would have a drastically large turning circle which could have resulted in the robot hitting the arena walls. Hence a 2-wheel differential drive with a caster wheel was chosen.

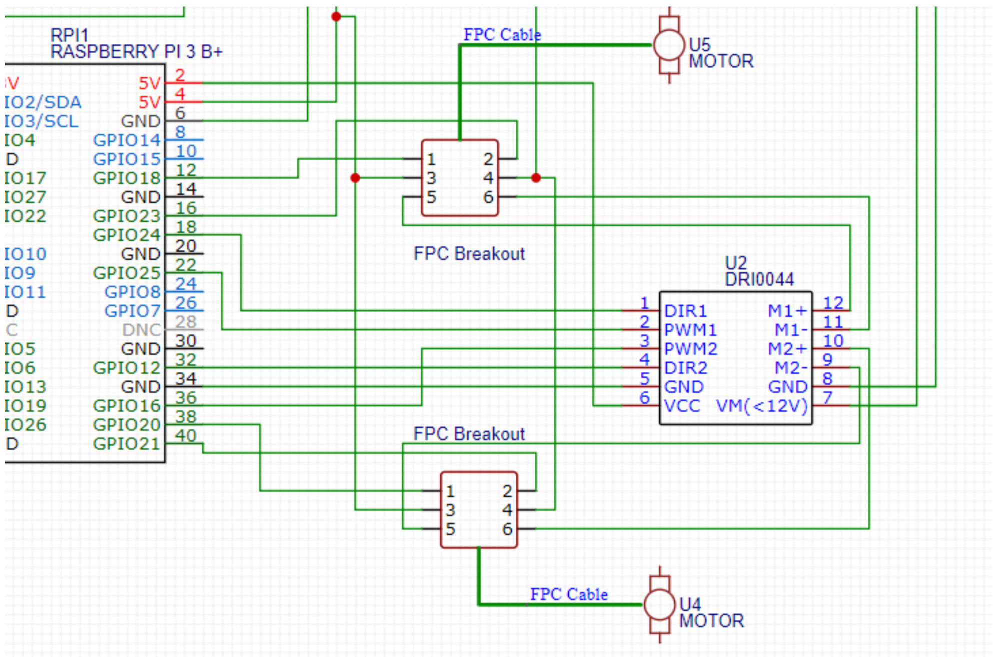

Figure 2.2.3.1 Schematic of propulsion system

2.2.4 Power Management Considerations

A goal for the electronic design was to minimise both the average power consumption (to extend the battery life) and the maximum power draw (to enable us to use lower spec'd components). Several methods were used to achieve this:

1. Moving the motors / robot slower for improved efficiency (this also has other advantages that were discussed elsewhere)

2. Connecting the motors directly to the battery to remove the inefficiency of the buck converter. Although the voltage of the battery pack can vary significantly throughout the discharge cycle, the peak voltage is still much less than the motor peak input voltage so this is safe to do so. Because the PWM based PI speed controller is equivalent to a switching mode power supply there are no issues with having a variable input voltage.
3. The electromagnet for the grabber is only energised when the robot is ready to collect a ball, so no power is wasted whilst idle.
4. The electromagnet and servo for the grabber only operate when the robot has stopped moving (ie: the motors are drawing 0 amps) so the maximum power requirements are decreased. The complex software algorithms are also paused whilst the robot is performing the grab collection of the target to minimise power consumption of the Raspberry Pi CPU.
5. The software algorithms plan the shortest path to the target to minimise unnecessary energy usage.
6. More efficient software algorithms were chosen in order to minimise the CPU energy consumption, such as utilising simple OpenCV detection for the targets rather than a neural network that does billions of operations for every image.

2.3 Software Design

2.3.1 Software Classes and overall design

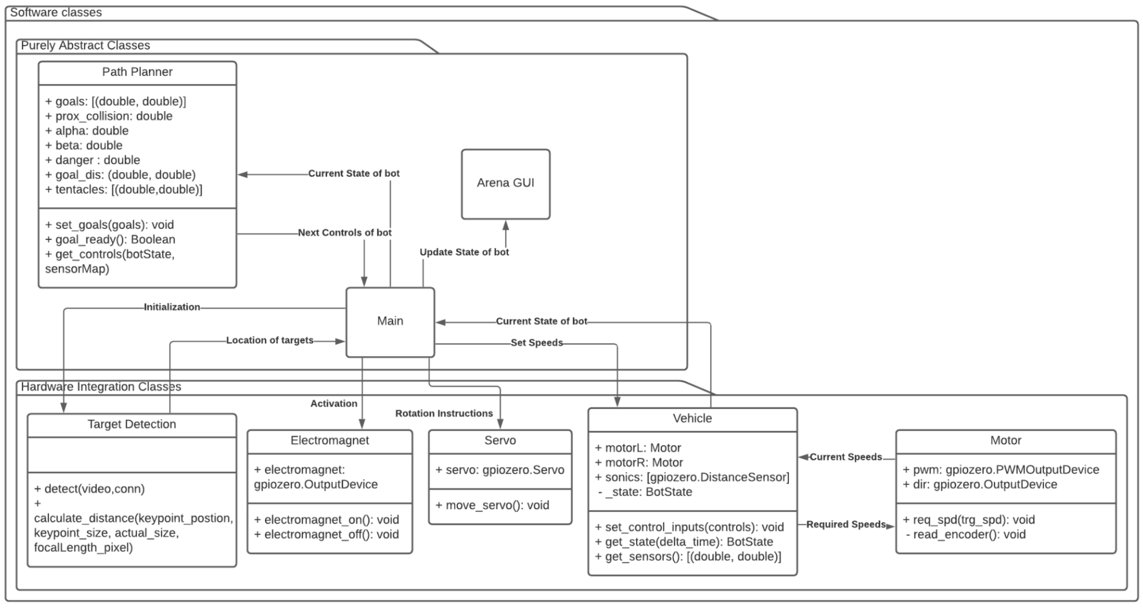

Figure 2.3.1.1 - Software Class Diagram

The software to control the robot was implemented following an object oriented style of programming. Each of the major tasks the robot was asked to perform were split into classes grouping commonalities and reducing dependencies between classes. The major classes were vehicle, target detection, path planning and arena GUI.

The vehicle class acted as the class closest to the hardware. It contained all hardcoded values pertaining to the robot such as pin allocations, sensor positions, wheel radii and track width. It also held functions to set the control speeds and read current speed of the motors, creating a current state for the robot. This class provided much needed abstraction for the rest of the system, encapsulating all physical aspects of the robot while providing functions to interact with the components for the higher functionalities.

The target detection class acted as a linkage between the camera and the robot software, it has a back-to-back connection with the path planning class. It returns the 3D position of the target with respect to the current robot position.

The path planning class acts as one of the highest classes. It deals with the path planning for the robot but with all values and actions highly abstracted from their physical meaning.

2.3.2 Path Planning/Obstacle Avoidance Algorithms

The path planning/obstacle avoidance algorithm focuses on 3 stages of movement. These stages all prioritize different locations to reach but plan paths to those locations based on the same method.

The first stage is the exploration stage in which the robot can visit a preplanned number of locations. When the last location is reached the robot will do a 360° spin. The objective of this stage was to explore the arena to detect any targets as the target detection algorithm would be running concurrently throughout all movement.

The second stage is the tracking stage which occurred when a target was found. The robot would then simply path to the found target updating its location as it achieved more accurate target detection data.

The final stage was the orientation stage, where the robot was close enough for pick up, it would orientate itself to allow the target capture system to reach the target. It would do this orientation by creating a new location, from the target and the surrounding obstacles at which the capture system could reach the target and path to this location.

The ultrasonic sensors were utilized to identify obstacles, their locations were then written to a circular buffer. This buffer would keep track of a number of older obstacle locations but would overwrite the oldest values with new ones as new values became available from the sensors. This helped to maintain a better picture of obstacles in the arena while minimizing localization errors as the slack in the gearbox shifted the internal memory of obstacles from their real world locations. A diagram representing how the circular buffer data structure was utilized in holding obstacle locations is shown in Figure 2.3.2.1.

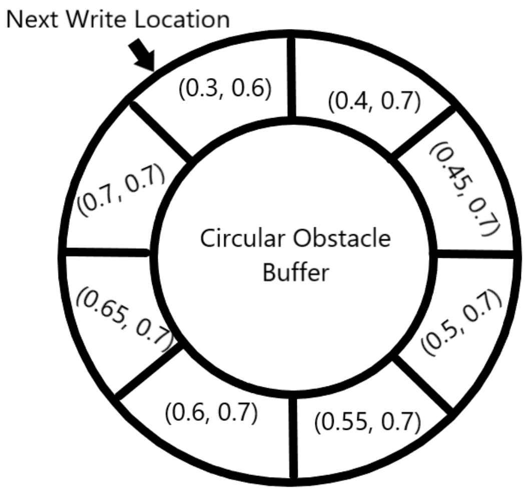

Figure 2.3.2.1 - Circular buffer diagram

Movement to each of the identified locations was achieved with a modified “driving with tentacles” strategy. This strategy simulates multiple paths that the robot could take, eliminating paths that would collide with obstacles. The algorithm then weights all remaining valid paths to choose a “best” path for the robot to take. The path with the lowest weight is then selected as the desired path to take. A flow diagram of the general algorithm is shown in Figure 2.3.2.1.

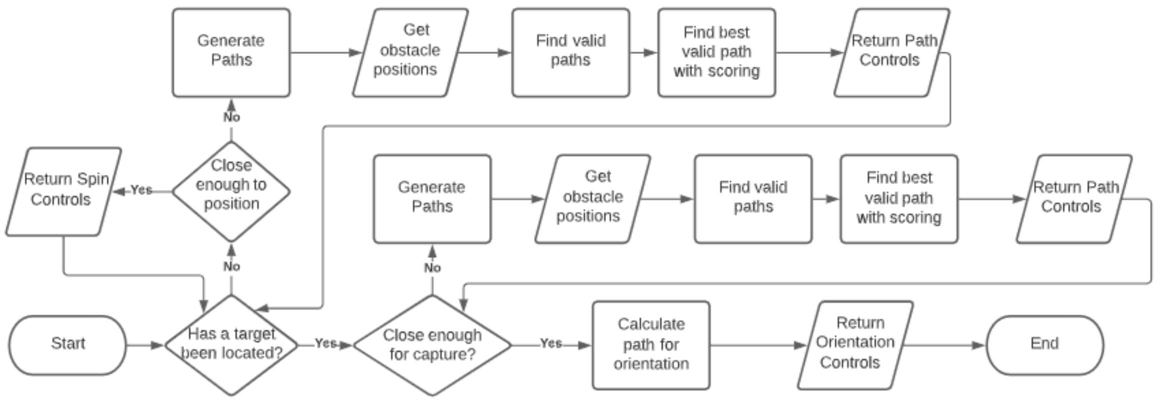

Figure 2.3.2.2 - Driving with tentacles flow diagram

The weight of each path was achieved by scoring the final position that path could achieve based on three scores: a distance score, yaw score and danger score. The influence of the scores in the weighting of the path could be increased or decreased with select variables within the class allowing for prioritization of certain scores over others when deciding the path. A diagram of the weights can be seen in Figure 2.3.2.3.

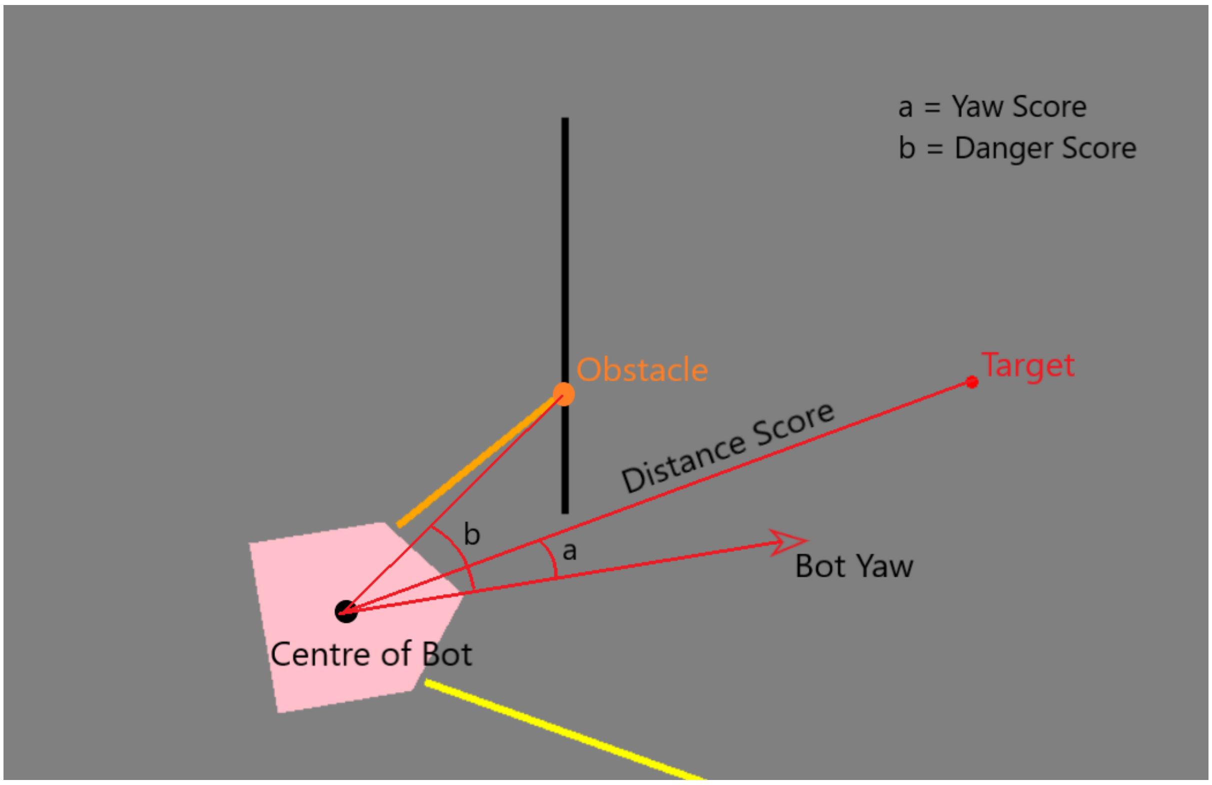

Figure 2.3.2.3 - Score Diagram

The first score was the distance score, associated with the alpha variable. This score was based on the distance of the bot from the chosen target. When the robot decreased the distance between itself and the target it's distance score would also decrease. This would encourage the robot to choose paths that got it closer to the target achieving the desired result of approaching the target.

The second score was the yaw score, associated with the beta variable. This score was based on the difference between the yaw of the robot and the angle from that yaw to the goal. When this difference was decreased the yaw score would decrease. This would encourage the bot to face the goal. This aided in navigation as it would face the ultrasonic sensors mounted at the front of the robot towards the goal, thereby identifying the obstacles in the way of the shortest path to the target.

The third and final score was the key innovation in the path planning algorithm, the danger weight, associated with the danger variable. This score was based on the degree to which the bot would face away from close obstacles. Specifically, it would try to minimize the difference between the robot yaw and the angle 90 degrees away from the aggregation of all close obstacle values. This was the weight with the most influence in the path planning algorithm. As such this would encourage the bot to preemptively face away from obstacles, allowing the bot to search out alternate routes that may increase the other 2 scores but will stop the robot getting stuck behind large wide obstacles. However, this would mean the robot would never prioritize getting to the target and instead prioritize obstacle avoidance. This was the beauty of the path planning algorithm. The circular buffer for the obstacle values would make the bot periodically forget older obstacle locations. This meant that the robot would turn away from obstacles and face parallel to them, however the initial obstacle locations would be forgotten, moving the aggregated obstacle position in front of the robot. The robot would then move forward to correct this. This would constantly occur until the robot stopped detecting the obstacle, meaning it had navigated around it at which point the bot would immediately refocus the target. This created a system in which obstacle avoidance and target prioritization could both be the primary factor in path decision making at different points all depending on the information received from the ultrasonic sensors. This allowed the bot to move around wide obstacles, capable of triggering all ultrasonic sensors at once by orienting itself away in search of longer alternate paths.

Together, the weighting system, circular buffer and staged prioritization design allowed for a dynamic and adaptable path planning algorithm, capable of exploring the arena, adapting to sensor inputs and preemptively searching for alternate pathing approaches.

2.3.3 Alternative Path Planning/Obstacle Avoidance Algorithms

There were two main path planning/ obstacle avoidance algorithms that were considered in this project. In the early stages of the design process the design of the arena and obstacles were unknown and therefore the path planning strategy that was most adaptable and resilient to different arena orientations and obstacle shapes was prioritized.

The first was a very simple solution, originally theorized to run on the square, high walled, campus arenas. One of the most simple approaches, it would simply move forward as its primary means of travel only considering alternate paths once the forward path was blocked. The alternate paths would be achieved by rotating the robot and then doing more forward motion. To move towards a target, the system would use the computer vision system to search for targets, either through parallelisation or at the start of each cycle, at this early concept stage both were a possibility. The robot would then turn towards the target and drive forward again towards the target. A flow diagram of this algorithm can be seen in Figure 2.3.3.1.

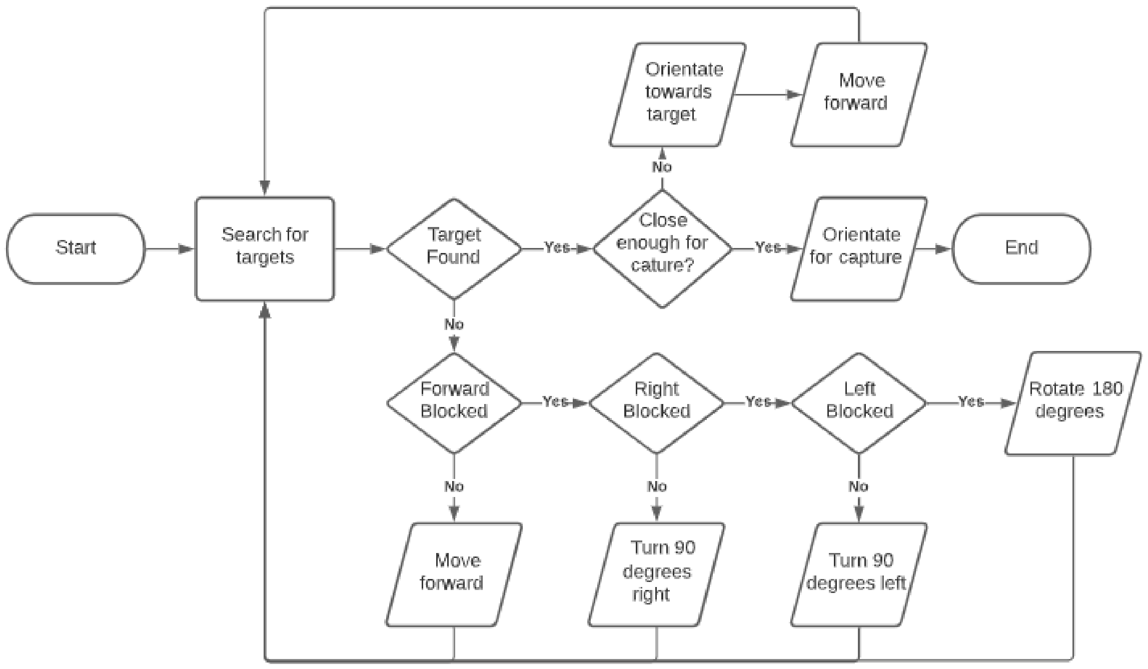

Figure 2.3.3.1 - Alternative path planning/obstacle avoidance algorithm

This design was not chosen for a number of reasons namely: rigid structure of the paths, lack of adaptability in path planning, possibility of loop between 2 dead end paths and the thought that it had a poor exploration pattern that would be unable to find balls in the centre of the arena. These reasons caused this path planning strategy to be abandoned even before an approach was programmed, instead the more adaptable "driving with tentacles" was chosen to be the basis of the path planning algorithm.

This led to the second major alternative considered. At this stage the "driving with tentacles" strategy was confirmed as being the most appropriate but it had the caveat that a large, plank like obstacle would be impossible to get around with the initial weighting system, that of only yaw and distance so a third weight needed to be used to path plan around these obstacles. These two strategies were the danger weight and the safe weight. The danger weight would be the eventually adopted strategy but the safe weight was also seen as a relevant and useful weight, considered in addition with the danger weight or in replacement of it. The idea of the safe weight was a weighting used to encourage the robot to move towards areas that it knew were safe with the obstacle detection. Areas that the robot did not detect an obstacle in for a certain distance represented "safe" areas and therefore they would be weighted accordingly to encourage the robot to move to areas without obstacles. This would encourage the robot to move around plank-like obstacles as parallel to the plank would be seen as safe areas.

This idea was held onto throughout the construction of the algorithm and implemented into the path planning simulation at which point the fatal error of the safe weight was discovered. The safe weight made the robot prioritize moving to "safe" areas, but these "safe" areas may be in the opposite direction to the target. Therefore the robot would be faced with a dilemma of going towards the target or the safe area. If the robot always prioritized the target, the robot could not navigate around the plank obstacle as it would cause the robot to go away from the target but if the robot prioritized the safe area, it would never reach the target, only moving towards it's known safe areas.

Therefore, the safe weight was not included in the final design and the danger weight was improved and tested extensively in the path planning simulation to make sure it could handle any number of differently shaped obstacles.

2.3.4 Target Detection/ Mapping Algorithms

The detection algorithm is implemented based on field respect with computer vision (CV). It is a subset of artificial intelligence that enables a computer to retrieve valuable information from images. In simple words, it enables the computer to see and observe.

To detect an object, there are several steps that should be performed. Most importantly, the feature (i.e. keypoint) extraction. After years of development and progress on this aspect, there is a wide range of choices available. For instance, the Harris operator and the FAST Feature Detector. Luckily, the target going to be detected in this project is circles in 2D perspective, it has relatively low difficulty to extract such a feature.

The detector used for this project is the blob detector from OpenCV. OpenCV is an open-source library of functions focused on real-time computer vision, which is widely used in the industry and has high reliability. It meets the initial criteria of our design: simplicity and robustness. The detection software will firstly apply a threshold to the input image, followed by the OpenCV blob detector to measure the response in the whole image plane. Since the actual size of the target and the focal length of the camera is all known, based on the detected target size in the pixel unit and the image coordinate of the target, the real coordinate of the ball bearing can be deduced by applying similar triangles. A diagram of these similar triangles is shown in figure 2.3.4.1.

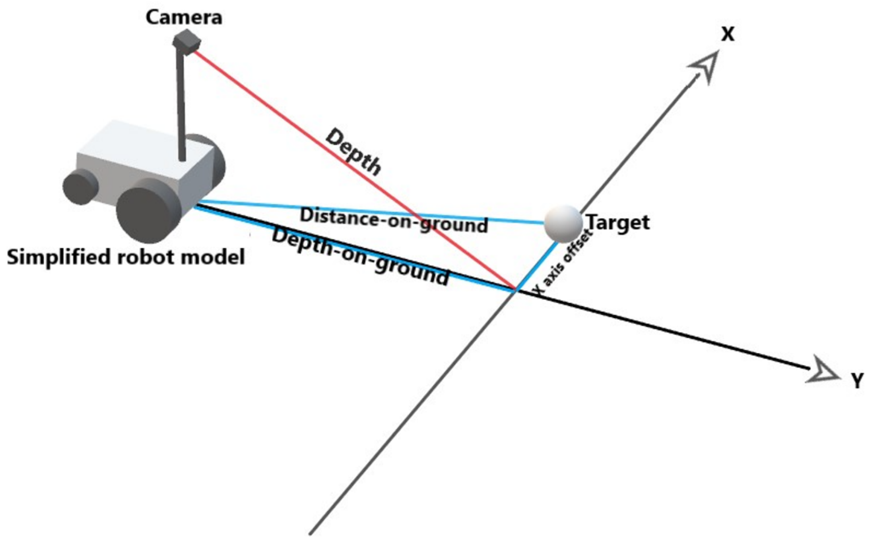

Figure 2.3.4.1-Mapping Geometry

Moreover, multi-processing has been implemented to reduce the overload of the detection process, which means the whole detection process is running separate and parallel to the rest of the system.

During experimentation, it was noticed that the nature of the silver ball bearings made it difficult to distinguish from the surrounding environment. Therefore, the final design has utilised two simple blob detectors working parallel. One of them is aiming for the main body of the silver ball bearings. And another one is aiming for the sparkling spot on the ball bearings. In most cases taken from different lighting conditions, the spot on the silver ball bearing has a much more prominent shine, and hence the potential targets can be further filtered by judging the distance between the main body and the sparkling spot. The detection accuracy of this strategy, after several trials, is above 95%. And the mapping error is within 1 square centimetre.

2.3.5 Alternative Target Detection/ Mapping Algorithms

The alternative design for Target Detection and Mapping algorithm also has high performance. Considering the illumination uncertainty of the final competition environment, our group has developed a backup detection/ mapping algorithm only based on target 1D information. (i.e. instead of requiring detected size and coordinates of the target, this algorithm can be functional with only the coordinate information)

The original algorithm detects both the main body and the spot (as we did in the primary design), now the alternative only detects the spot on the bearing and totally ignores the body of the bearing. For the mapping stage, the algorithm will need to have the information about the camera angle. By projecting the real word to the image plane and with the calculated pixel offset, the actual viewed angle of the target can be calculated. And since we know the camera height, the distance of the target can be calculated by the Pythagorean theorem. The performance following this alternative is close to our primary design.

3.0 System Development

3.1 Strategies

The general strategy for the target capture as performed by the robot is as follows. The robot initially begins by initiating its algorithm. It initializes the parallel running of the target detection and movement/capture programs. The path planning algorithm begins the robot in an exploration stage, visiting predetermined locations in search of the target. Meanwhile the encoder reports the speeds of the wheels to keep track of the position of the robot in relation to the rest of the arena. The robot will visit each location one by one, using the localization system to determine when it has reached the desired locations. Once it has reached the final location it will spin in an attempt to find any other targets.

If any target is located in this exploration stage by the target detection system it will cause the path planning system to enter the tracking stage. The target detection system will calculate the position of the target relative to the robot and use the current robot position to find the global location of the target. The path planning system will then be notified of the new target location and try to move towards it. During this movement the target detection position will constantly recalculate the position of the target location causing it to always remain accurate relative to the robot.

Once the robot is close enough to the target it begins the orientation stage. The begins a predetermined movement based on the position of the capture system to orient itself to be able to capture the target. Once the robot is in position, it powers the electromagnet and actuates the servo to capture the target. This involves rotating the servo a set amount causing the electromagnet to lower enough to be one top of the target. From this point the magnetic target stays attached to the electromagnet. The servo is actuated again to move the arm back to its starting position. This completes the general strategy of target capture as performed by the robot.

3.2 System Integration and Testing

As discussed previously, due to the forced separation of the team both the software and hardware elements of the robot were split up into modules. The communication between the different blocks is shown in Figure 3.2.1.

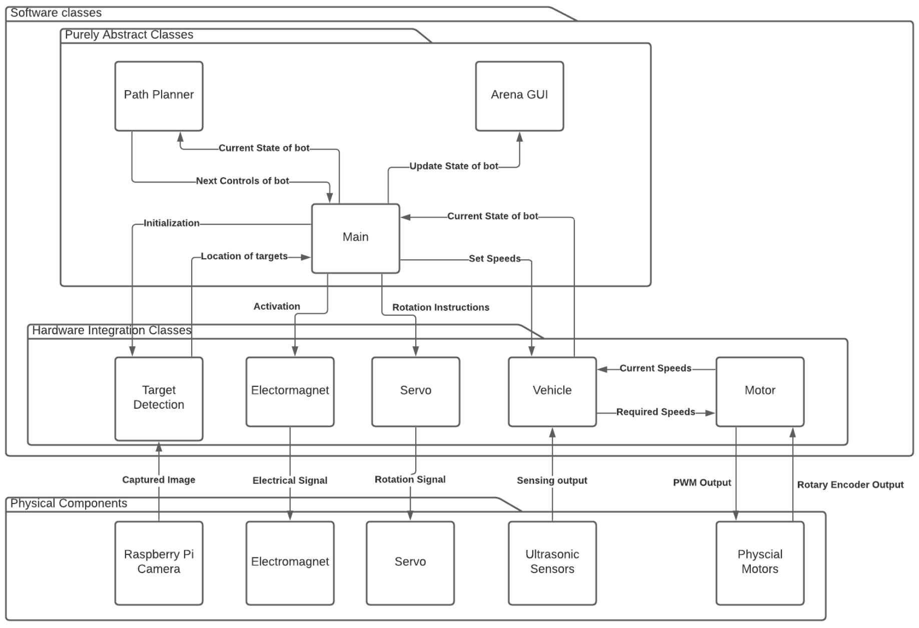

Figure 3.2.1 - System block diagram

Although the split, modular design made it easier to work on in parallel, it also increased the testing needs of the robot. A key innovation for the software testing side was to create a simulation environment for testing purposes by creating "dummy" versions of each of the various blocks. This made it trivial to swap out the ultrasonic sensor class (that communicated with GPIO) to another that simply simulated the distance sensor outputs. This meant that absolutely nothing needed to be changed within the path planning class to accommodate this.
In order to facilitate the quick analysis of problems when testing components, a GUI was also created, as shown in Figure 3.2.2 .

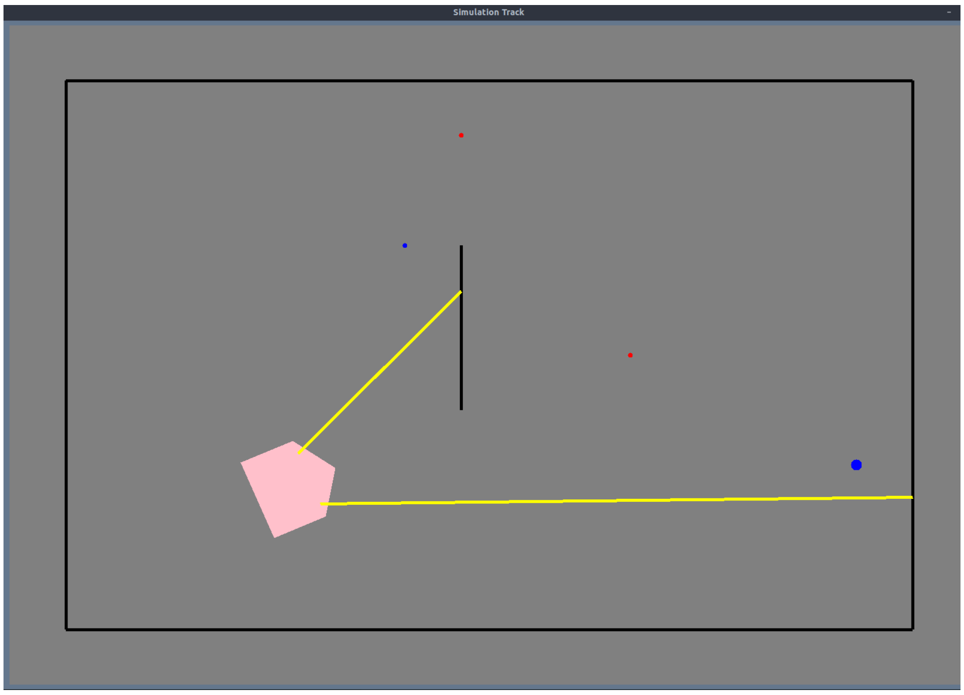

Figure 3.2.2 - Simulation environment

Test setup 1: The first test environment relies heavily on the aforementioned simulation environment. The path planning was our most complex piece of software and as such, it was critical to validate that it worked in many different scenarios. Using the simulation we were able to start the robot in various positions and orientations, with different obstacle, distractor and target locations. This allowed us to increase the robustness of our path planning algorithm and ensure there were no edge cases that would cause it to fail.

Test setup 2: The second set of tests also relies upon the simulation environment. Different permutations of sensor configurations were run in order to determine the optimal number and angle of the ultrasonic sensors. This testing made it clear that some mounting locations could enable the edge of the robot to hit the obstacle whilst being completely blind to it. The use of this testing enabled us to avoid allowing that issue to occur in real life.

Test setup 3: The robot localisation system relies on knowing the equations of motion for the vehicle and the rotation of the motors. This rotation is given to us by the encoder on each motor, however they are not perfect and testing demonstrated systematic error in the measurements. In order to test them the robot was commanded to move 1 metre forwards and the error correction factors (a simple linear scaling for each encoder count) was updated based upon this. This test was repeated until the robot was able to achieve a mean movement distance of 1 metre. However, a standard deviation of ~15% still remained. This was deemed to be an acceptable level of error, as the continuous detection of objects would act as a feedback loop to correct the position error.

Test setup 4: In a similar manner to the previous test, the rotation accuracy of the robot needed to be verified. To verify the behaviour the robot was commanded to turn and then go to a position at a certain angle. The error from this was used to tune the motion model's track width of the vehicle, and thereby increase the localisation reliability.

Test setup 5: Measurement of objects at various distances demonstrated that the ultrasonic sensors were found to have a high accuracy in the necessary range of 0.1-1.5 metres. However, this accuracy was found to significantly degrade when the detected surface is not perpendicular to the signal and the object may not even be detected at all. This information was used to help analyse the simulation results of possible mounting locations for the sensors, and therefore make object detection more robust.

Test setup 6: An important facet of the camera detection system was the translation of a detection in pixel coordinates to the 2D plane of the robot's real environment. Our translation system relied on geometric transforms using trigonometry, however it is also possible that the NoIR camera lens introduces distortion that skews results. Target objects were placed in various locations on the plane and the locations were calculated using the camera system. The ground truth locations were then compared to the estimated positions and the parameters of the algorithm were adjusted to produce a more accurate result.

3.3 Project Management

3.3.1 Team Roles

The team was separated into distinct subsystems to handle specific problems in the robot design, construction and execution. These roles are expressed in Table 3.3.1.1.

Table 3.3.1.1 - Team role allocations

Name	**Roles**
Mihin Perera	Propulsion/Power Management
Jack Coleman	Simulation/OS/Chassis Construction
Zhaoning (Steve) Lu	Computer Vision/Mapping
Sean Martin	Path Planning Scrum Manager/ Management
Cameron Hancock	Arm Design Final Construction

Due to the lockdown restrictions placed on Melbourne at the time of the project, Zhaoning and Sean could not access the robot without courier or package delivery as Zhaoning was overseas and Sean lived outside of the exercise range. Therefore, it was decided that Steve and Sean should focus on higher level software development while the rest of the team, including Mihin, Jack and Cam would work on some physical level to aid in the robot construction. This made the construction of the robot much easier as the load was split among multiple members of the team that were within range to physically exchange parts creating an efficient and streamlined construction process.

The allocation of specific subsystems to each member of the team allowed for each member to do in depth research about their system and if there were any problems within that system it was very clear who was responsible for the construction and should be contacting when correcting the subsystem.

The allocation of management ensured weekly responsibilities like the weekly progress report were taken care of and the larger picture of the project and timeline were kept in mind as the weekly sprints were in progress.

Together these allocations aided in the timely completion of the project by creating a system in which each member of the team was clear on their responsibilities and deliverables for the shorter sprint cycles while the overall picture was still kept in focus with the weekly progress meetings.

3.3.2 Critical Path

The critical path is the most important steps or tasks required to complete this project. The preliminary plan of the timeline of task completion is shown in Figure

3.3.2.1. However, there were deviations from the proposed plan, both due to systematic and unforeseen errors.

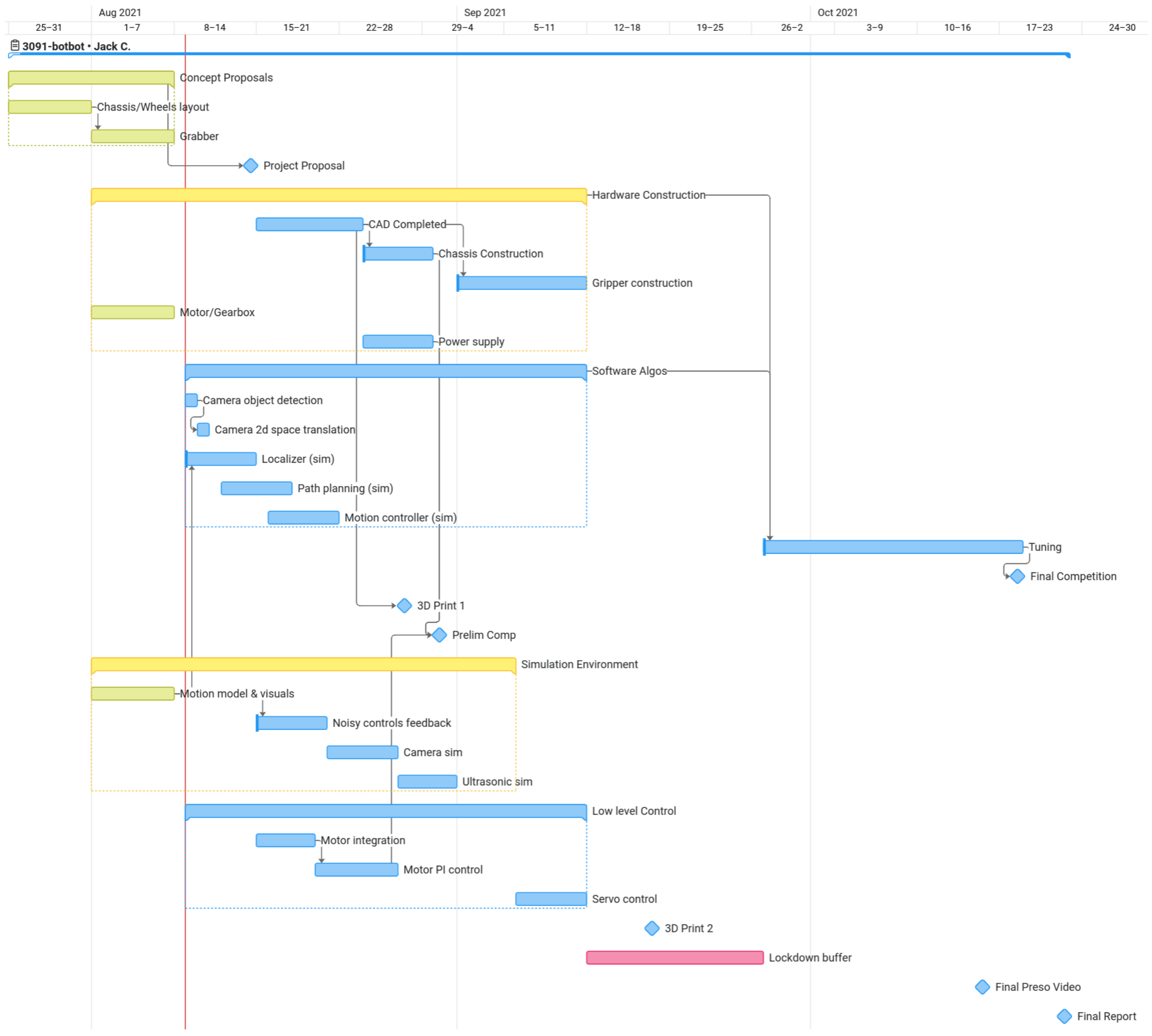

Figure 3.3.2.1 - Preliminary Project Timeline

This project began with the concept and design phase chosen to be done within the first 2 weeks. This involved each member of the team being designated roles and researching to come up with solutions for their own subsystem attached to their role. These were discussed at length within the team, multiple different designs were suggested and the best subsystems were chosen to be used in the project. These made up the project proposal as a collection of these different subsystem designs and a simple way in which they would be connected. In reflection, this was an appropriate way to get an in depth view of each subsystem, however the integration of these subsystems was often overlooked, leading to integration problems later on.

In parallel to the report each individual subsystem was being worked on. These subsystems were constructed in parallel by each member of the team. The simulation software was set up early in the project. This allowed path planning/obstacle avoidance, computer vision, grabber, CAD designed ultrasonic sensor mounts, motor and chassis design to be performed at the same time. However, this approach raised some problems. Although path planning/obstacle avoidance and computer vision were easy to work on concurrently as only the software was required, physical construction encountered a number of delays and problems.

The original plan involved splitting the components so each member could work on their subsystem independently. The motor components, raspberry pi and the rest of the components were each split among team members. Lockdown restrictions were announced that week leaving the components stranded across 3 households. This meant that when the motor system construction testing or ultrasonic testing was needed they could not be completed as the raspberry pi was in a different location. Fortunately, the team found a way to pass along components in a chain eventually getting all the components in a single location. Reflecting back on this part of the project, it became clear that it was essential to keep all components together and with a single person to minimize time delays in transporting components and to ensure that any constructed system could be immediately tested to stop cascading errors.

Once all the hardware components were together chassis construction could commence with the already completed motor system used together with the raspberry pi and the rest of the kit to produce the preliminary design.
The power supply construction was pushed back to after the preliminary competition. This was because it was identified that the preliminary competition could be run with a direct power connection rather than battery powered. Therefore, to minimise risk in the preliminary competition and to streamline the construction process it was decided that the initial construction be done by Jack, with power supply implementation done by Mihin to share the load of the construction process. This meant that the power supply construction would be pushed back to allow Jack to hand off the robot to Mihin in relaxed restrictions.

Once all the hardware was constructed the integration process with the software could be completed. In reflection this was one of the most seamless parts of the project. The adaptable path planning software meant values could be changed to accommodate the large error of the gearbox system leading to an effective preliminary design that completed its task easily and with flying colours.

After the completion of the preliminary competition the other hardware elements, such as the camera, power management and grabber needed to be added to the design. Reflecting back, the initial construction was done without much oversight or consideration for the final design. Each member of the group was so focused on their own section and the overall timeline, that the chassis design and layout was often overlooked causing the grabber design to suffer in the end. The camera mount was created by Jack and then the entire robot kit was handed to Mihin for power management construction. Mihin completed the power management, attaching the power pack and redistributing components to make a compliant robot which was then handed to Cam for the grabber integration and final competition execution.

Cam was to implement his grabber design onto the robot, however due to the layout design no area had been allocated for the grabber design and the servo had not been tested with the attached wood and electromagnet. This late testing stage led to a number of discoveries that would drastically affect the robot's design. It was discovered that the servo did not have enough supplied torque to move electromagnet on a long arm. Therefore, the arm had to be shortened to accommodate the electromagnet which forced the arm to be attached to the side of the robot, the spot left free by coincidence. Furthermore, the robot had not been tested since the preliminary competition at this point, leading to a number of issues setting up the robot again with a different computer system.

Reflecting back on this period, it was extremely difficult to balance the consistency of elements in making the robot functional, while also sharing the load of construction. Each time the robot was passed over it took a process of explanation and direction to set up the robot each time, often accompanied by errors caused by miscommunication or unfamiliarity with the robot layout. This was an unfortunate effect of the lockdown restrictions,with in person collaboration restricted, each time testing had to be completed either the robot had to be passed to a member with an existing setup or a new testing setup had to be constructed.

Once Cam had set up the testing environment the camera detection and target capture testing was conducted. After these subsystems were tested the integration of all subsystems together was conducted, leading to the final competition robot. Reflecting back, the setbacks experienced before led to integration of these systems to occur very close to the final competition date. This caused a large amount of crunch work within the group to allow the robot to function on competition day. This could have been improved with appropriate planning for unforeseen setbacks earlier allowing more breathing room close to the final competition. An overview of the critical path can be seen in Figure 3.3.2.2.

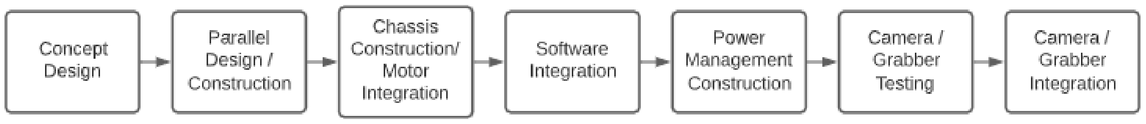
Concept Design
Parallel Design / Construction
Chassis Construction/ Motor Integration
Software Integration
Power Management Construction
Camera / Grabber Testing
Camera / Grabber Integration

Figure 3.3.2.2 - Critical path flow chart

Reflecting on this critical path, it is abundantly clear that more planning of how the hardware components would be placed on the robot chassis and how they were to interact was needed. It was an unfortunate effect of limited physical access to the robot but the physical state of the robot was often lost in communication leading to critical compromises in the robot design ultimately resulting in the inability to capture the target in the competition.

3.3.3 Project Budget

Table 3.3.3.1 - Project budget

Part NB: Hyperlink the part to supplier	**Qty**	**Total Cost**
Raspberry Pi4 8GB	1	$126.56
Raspberry Pi Camera Board v2 - 8 Megapixels	1	$34.57
Busterlite MicroSD Card - rasbian	1	$18.45
UBEC DC/DC Step-Down (Buck) Converter - 5V @ 3A output	1	$19.70
MP000334 Battery Holder, Flat, Wired, 6 x AA	1	$3.15
TowerPro SG90C 360 Degree Micro Servo (1.6Kg)	1	$8.06
Tamiya 70168 Double Gearbox Kit	1	$18.40
Tamiya 70144 Ball Caster Kit (2 casters)	1	$10.90
Replacement motor with shaft encoder	2	$5.24
Tamiya 70111 Sports Tire Set (2 tires)	1	$13.95
2x1.2A DC Motor Driver (TB6612FNG)	1	$7.94
HC-SR04 Ultrasonic Module Distance Measuring Sensor	2	$7.70
FPC/FFC flat cable transfer plate 1.0 mm pitch 6 pins	4	$4.48
FFC cable same side contact 1.0 mm pitch 6 pins 30 cm length	4	$1.39
40 way header pins	6	$14.10
10 sets M3x6 screw low profile hex head cap screw	1	$1.09
Tamiya 70157 Universal Plate Set (2pcs.)	1	$12.60
10 sets M3 * 20 nylon standoffs	1	$3.69
Electromagnet	1	$10
Wood	1m	$8
	TOTAL	$329.97

Table 3.3.3.2 - Estimated budget vs project budget

Estimated Budget	Project Budget
$407.91	$329.97

The actual project budget was significantly less than the estimated budget. This was due to several of the components in the preliminary being unnecessary in the final design. These included only 2 ultrasonic sensors being necessary for obstacle avoidance. The wall jack was also unnecessary as the final competition design ran purely off the battery power pack. Furthermore, the epoxy and miscellaneous budget allocation were not used due to alternate approaches to the construction process.

4.0 Recommendations

A number of recommendations can be made for the robot:

1. Soldering all of the electronic components (propulsion and power system) onto a specially designed PCB. This will increase the robustness of the electronics and enable a smaller chassis to be developed for the robot
2. Redesigning the grabber mechanism to allow a larger margin of error in the path-planning and grabbing the target
3. Upgrading reading system for the shaft encoders to have higher accuracy at higher speeds. Ideally this (along with the speed control for the motors) could be moved off the raspberry pi onto something real-time capable. This would allow the robot to move at significantly faster speeds whilst also increasing the robustness of the odometry.
4. The software system, although functional and clear, could be improved through additional refactoring. In the software code there were a number of points where hard coded values or movement planning was used. For example, the simulation of paths in the path planning was independent of the actual movement code in the vehicle class. With some refactoring this could be streamlined to increase the ability to extend or modify the code.
5. The software algorithms could be streamlined to reduce computation time. Currently the algorithms tend to be independent within stages causing multiple calculations of the same values, for example distance to the robot being calculated multiple times. With efficient data structures these calls could only be performed once per distance call. This would free up computational resources of the Pi and allow for more effective real time processing on the robot.
6. A small collection bucket could be constructed underneath the grabber arm to capture balls that have already been collected. The balls collected from the electromagnet arm could be dropped by depowering the electromagnet causing the ball to drop from the arm. This could be effective in allowing multiple balls to be captured by the robot and carried in its collection system.

5.0 Conclusion

The robot proved to be very capable in the final competition, being able to accurately identify, and move towards the target bearing consistently as a result of robust path-planning strategies, computer vision methods and a unique exploration strategy within the arena. It was able to accurately estimate its own position in the arena through the combined use of the ultrasonic sensors and odometry and an effective power management system meant that the robot was able to spend extended periods of time moving independently. The relatively simple modular design allowed the robot to be designed quickly and effectively despite the limitations of COVID-19, with the robot being passed around to the majority of the group members (who were able to receive it) to implement their designed systems. The only system requiring improvement is the grabbing mechanism as the robot was unable to collect the bearing within the competition setting. Despite this, overall the robot accurately met most of the requirements of the design brief, incorporating a number of key innovations that resulted in the robot being very effective in the competition, being a culmination of a multidisciplinary approach to robot design.