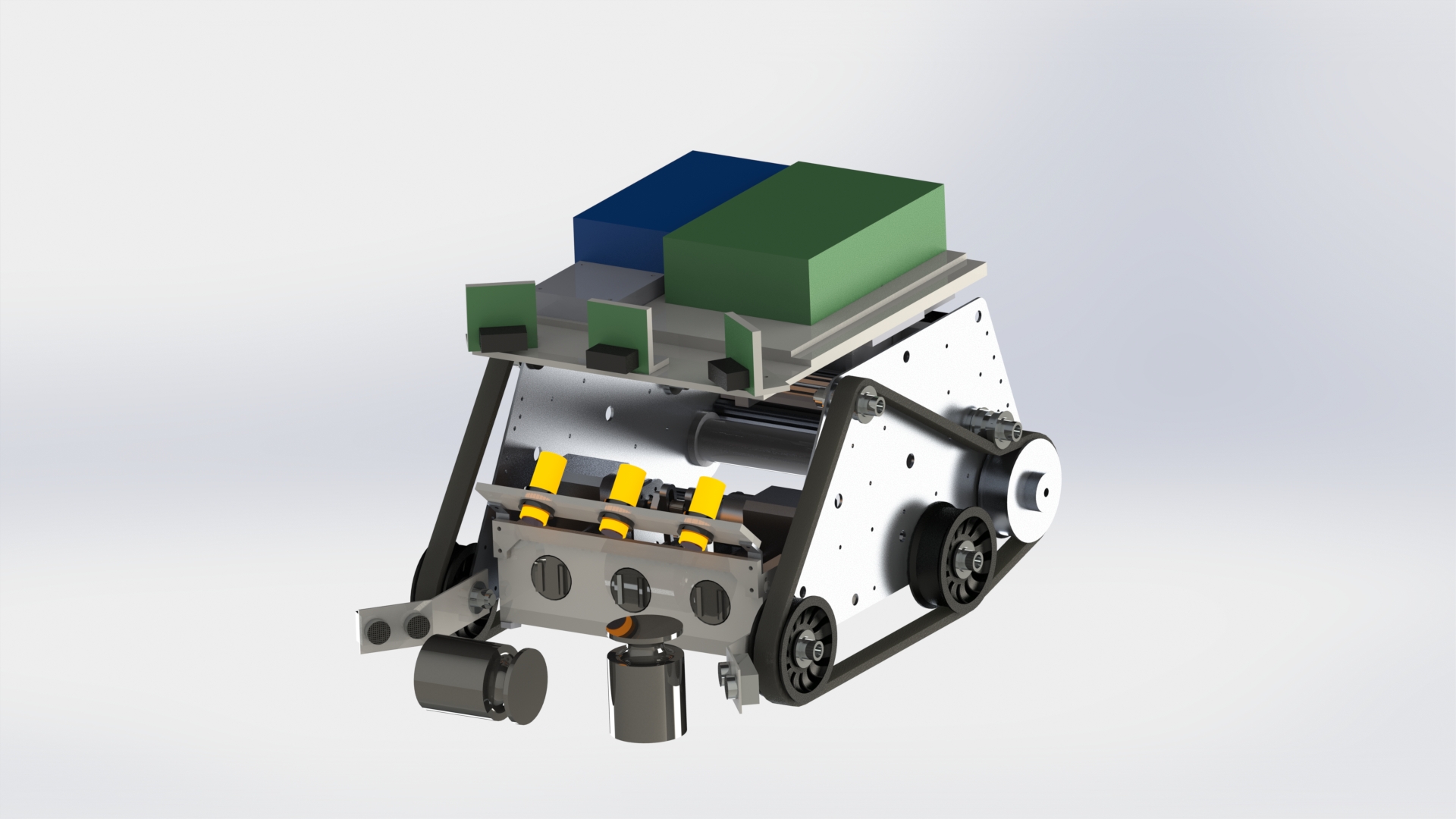
University of Canterbury

Robocup Progress Report 2

Group 10

Jack Hendrikz  
Peter Nicholls  
Ryan Taylor

21 August 2015



**Table of Contents**

[0.0 Executive Summary 2](#_Toc427919106)

[1.0 Introduction 2](#_Toc427919107)

[2.0 Mechanical Design 4](#_Toc427919108)

[2.1 CHASSIS 4](#_Toc427919109)

[2.2 LOCOMOTION 4](#_Toc427919110)

[2.3 PACKAGE COLLECTION 4](#_Toc427919111)

[3.0 ELECTRICAL DESIGN 8](#_Toc427919112)

[3.1 Circuit Boards 8](#_Toc427919113)

[3.2 Sensors 9](#_Toc427919114)

[4.0 SOFTWARE DESIGN 12](#_Toc427919115)

[4.1 Program Structure 12](#_Toc427919116)

[4.2 Support Functions 12](#_Toc427919117)

[4.3 Top-Level Design 13](#_Toc427919118)

[4.4 Mid-Level Design 13](#_Toc427919119)

[4.5 Low-Level Design 15](#_Toc427919120)

[5.0 Design Evaluation 16](#_Toc427919121)

[5.1 General 16](#_Toc427919122)

[5.2 Identification 16](#_Toc427919123)

[5.3 Movement 17](#_Toc427919124)

[5.4 Collection 17](#_Toc427919125)

[5.5 Construction 17](#_Toc427919126)

[5.6 Safety 18](#_Toc427919127)

[6.0 Future Development 19](#_Toc427919128)

[6.1 Mechanical Development 19](#_Toc427919129)

[6.2 Electrical Development 19](#_Toc427919130)

[6.3 Software Development 20](#_Toc427919131)

[Contribution Statements 20](#_Toc427919132)

[Appendix 1 – Matlab Code and Derivation 22](#_Toc427919133)

[Appendix 2 - Circuits 24](#_Toc427919134)

[Appendix 3 – Package Detection 26](#_Toc427919135)

[Appendix 4 – Efficiency Tests 26](#_Toc427919136)

[Addition as Polar 26](#_Toc427919137)

[Addition as Cartesian 26](#_Toc427919138)

[Appendix 5 – Bill Of Materials 2](#_Toc427919139)

[Bill Of Materials (Robot) 2](#_Toc427919140)

[Bill Of Materials (Pick Up Mechanism) 2](#_Toc427919141)

[Bill Of Materials (Circuit 1) 3](#_Toc427919142)

[Bill Of Materials (Circuit 2) 3](#_Toc427919143)

[Bill Of Materials (Circuit 3) 3](#_Toc427919144)

# Executive Summary

This report discusses the design of group 10’s robot that will be entered in the Robocup challenge in October this year. The robot has already performed well in the initial functionality assessment, achieving a perfect score.

Three modules are presented with a detailed overview in this report. The mechanical system module includes the robot chassis, locomotion method and package collection. Next the electrical system module includes an evaluation of sensors used, their position on the robot, and the circuit used to process their resulting signals. Finally the software module which includes all the programming required to allow the robot to function autonomously.

An evaluation of each module is conducted to contrast the functionality against the robot’s requirements outlined in the initial report. The report also contains a future development section that discusses the potential improvement that can be made to the robot before the final competition.

The robot has current completed 16 of the 25 requirements and performed well in the functionality assessment. It can detect objects such as walls but has some issues with pole detection. It can identify, move to, and collect weights around the arena however there is much to be improved. The code allows for wall following but pathfinding is yet to be implemented. Overall the robot is well on its way to performing as specified on the test date and will be significantly improved over the following months.

# Introduction

In this year’s Robocup Zombie apocalypse, teams have designed robots that search autonomously for food packages. The robot will compete in the arena against another robot for these packages. The battle will take place in an unknown environment which includes immovable obstacles, walls, and a series of rough sections of ground all unknown to the robot. The robot will need to return the food packages to the team’s headquarters. These food packages are represented by cylindrical weights made from steel, and will be placed in both easy and hard to reach places.

This report takes our chosen design from Conceptual Design Report (CDR), and explains the technical details on how the robot is built and how it will complete the tasks. As a mechatronic system, the details are split into mechanical, electronic and software design, each of which are crucial for an effective, autonomous robot.

# 2.0 Mechanical Design

## 2.1 CHASSIS

The design will use the chassis supplied as it will take too long to design and construct a better one. The one supplied gives good mounting holes for the wheels in various positions.

## 2.2 LOCOMOTION

Locomotion is an important job, having to do a range of tasks. When designing the wheels you have to keep these things in mind, maintaining the tracks on the wheels, getting enough traction on the track from the motor, having enough ground clearance and how much and where the tracks touch the ground. The drive motors will be two 12V Low noise 28PA51G DC motor, these will be located in the designed position on the chassis wall.

Opting for 3D printed wheels allowed the ability to have control over all these parameters. 3D printing the wheels allows to have them designed in Solidworks and modelled appropriately to meet the parameters. Having the correct lip on the edge of the wheel is essential or else the track may come loose, this was modelled and a suitable lip designed. To get traction from the motor pulley to the track relied on getting the greatest amount of surface area. Having 180 degrees of contact with the pulley allowed enough traction. The value of 30mm was set as ground clearance, this set the radius of the wheels as the mounting holes are also set. To keep the area the track is on small will improve performance when moving, the downside is a loss of stability, and this was taken into account when choosing how much length of track would touch the ground. A middle ground was chosen with a slightly smaller length of 200mm, than the kit was designed with length of 250mm. Solidworks showed the centre of mass of the robot in the centre of the track length which is ideal, seen in Figure 1.

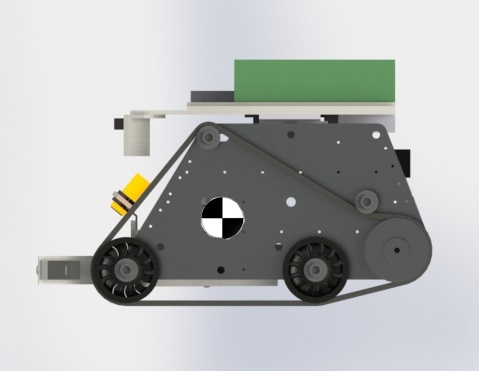


Figure , Shows the side of the robot with the wheel design and the centre of gravity.

## 2.3 PACKAGE COLLECTION

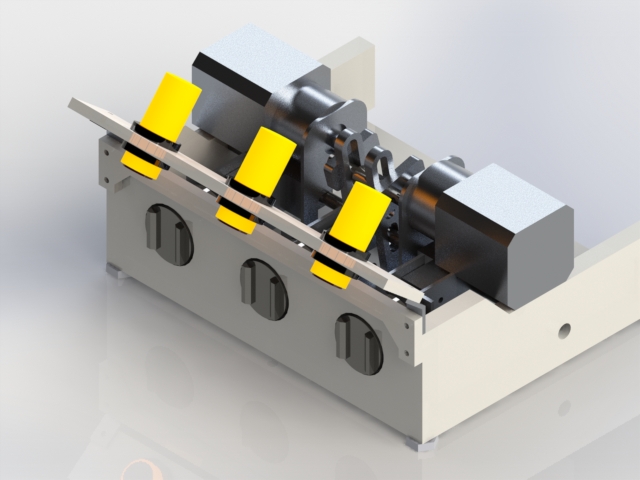
The mechanical element of the build requires a system to pick up and store three packages. The main principle for our design will be using magnetic force to pick and store packages. Three strong magnets in a row along the front of the robot will be ready if any packages are detected to pick them up. Once three packages are picked up then the robot will drive back to base and drop them off by retracting the three magnets though the perspex barrier. The force of the three magnets requires a greater force to retract them. This greater force has to overcome the 450N from the attraction between the magnets and packages. To get this force two geared stepper motors, FIT0349, will be used in junction with a reciprocating mechanism. This mechanism changes rotary motion to linear motion and also gives a step up torque simultaneously. Figure 2, shows the overall mechanism and Figure 3, shows more detail of the moving parts.

Figure , Shows the overall design for the pickup mechanism.

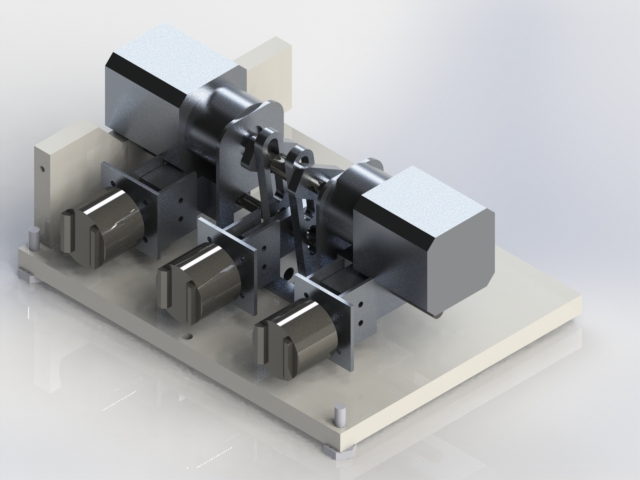


Figure , Shows a stripped away picture of the pickup mechanism.

Linear motion is provided by a simple slider crank idea, see Figure 4. Two geared steppers are mounted facing each other on a bracket holding them in line for direct drive. The two geared steppers are needed to provide enough torque. The geared steppers have an eccentric shaft between them, this gives the torque. The eccentric shaft can slide up and down a link, the other end of the link is pinned on the bottom of the mechanism. Down the link a rod, parallel the eccentric shaft, is restricted to horizontal movement. A connection is made between the three magnets and the rod. The magnets are mounted in the front held in by a perspex barrier. When the link sways back and forth it push and pulls the rod thus moving the magnets in and out. All the custom parts will have to be machined.

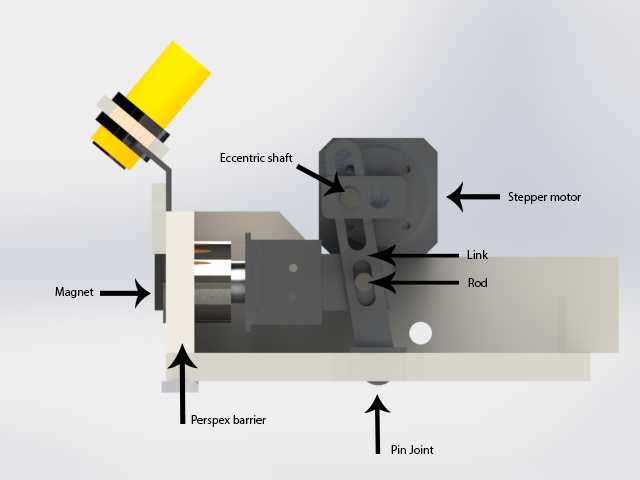


Figure , Shows a side view of the pickup mechanism with labels showing different parts.

Below is an exploded view of all the part for the pickup mechanism, Figure 5.

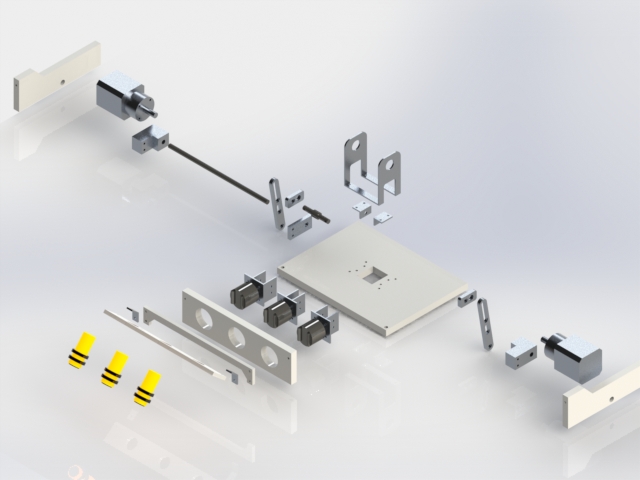


Figure , Shows an exploded view of the pickup mechanism parts.

Making sure the packages fall off when the mechanism is operated is essential to the success of the robot. The mechanism uses two stepper motors with a torque of 1.78 Nm each. To make sure this will happen, we calculated in Matlab the force that will retract the magnets, seen in Figure 6. The blue line shows the force needed to release the magnets and the red line is the calculated force applied to the magnets. The packages will be dropped off between 80 to 160 degrees. This is a small band of the force curve that the figure shows will be enough force to drop the magnets off. The reason the second peak of the curve is lower than the first is because when the eccentric shaft is on the lower side of the turn less torque is transferred. See Appendix 1 for the code and hand calculations. Mechanical systems aren’t 100% efficient and this calculation doesn’t take this into account, the force produced will be less than is shown in the real world.

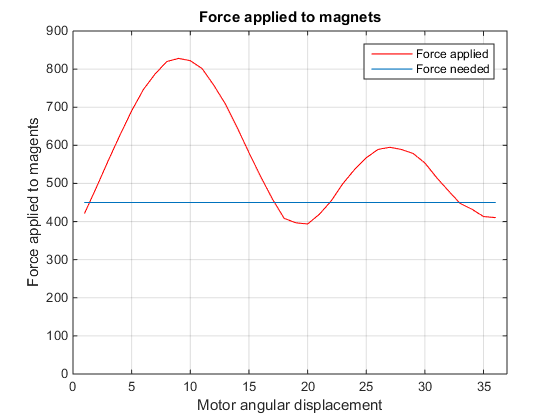


Figure , Shows the force produces by the pickup mechanism applied to the magnets.

The design focus’ on overcoming big problems that could put a halt to package collection, this consisted of different orientation packages (fallen over), packages tight corners, fast and simple package collection. The main design feature we want to cover is the fallen over packages. If the other team knocks the packages over then then the packages can still be recovered and if the opponent can’t recover fallen over packages then it will be an easy win. Secondly having the two of the magnets off centre of the robot will allow for corner collection easier. Finally having our collection being fast, once contact with the package is made it won't come off. Making the collection reliable and consistent.

The whole pickup mechanism tilts up and down to allow for when the robot drives around and goes into the opponent's headquarters. Having the mechanism tilted up will not allow any package to be collected. When picking up packages the mechanism will tilt down to connect on the package, then once the robot has driven into the package and confirmation the package is on board will raise the mechanism. Tilting will take 5 to complete the collection so will be a total of 10 second.

# 3.0 ELECTRICAL DESIGN

The electrical design of our robot consists of two modules, the arrangement of the sensors to enable detection of the robot’s environment and the boards used to control the signal outputs to these sensors.

## 3.1 Circuit Boards

In order to drive the sensors and motors with the correct voltage and to receive the desired signals for the sensors, the provided circuit boards must be implemented into the robot design. The circuit boards consisted of **Insert what circuit boards we used**. After some discussion we concluded that they should be mounted on the underside of a perspex sheet (below the Arduino). **Figure 7 below highlights this arrangement.**

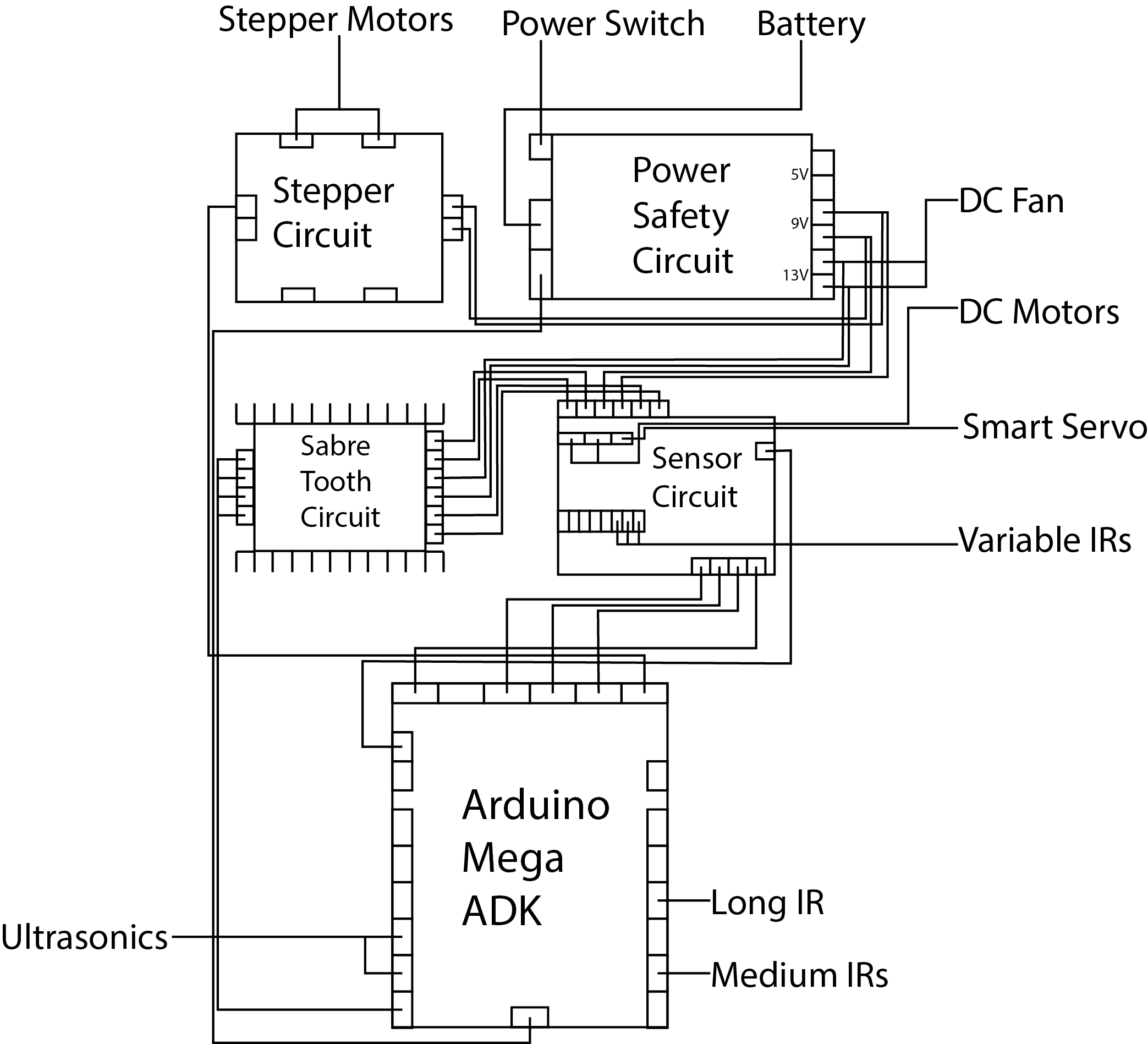


Figure , Shows the overall electrical circuit board connections.

This allowed the boards to be arranged in a compact design that would be easily accessible through a quick release mechanism that allows the mount to swivel. The boards were arranged in such a way that they could be connected easily to one another as well as the Arduino. The cabling could be neatly run without the risk of interference with the pickup mechanism or the risk of potential damage by being exposed externally on the robot. The quick release design is shown below in figure 8.

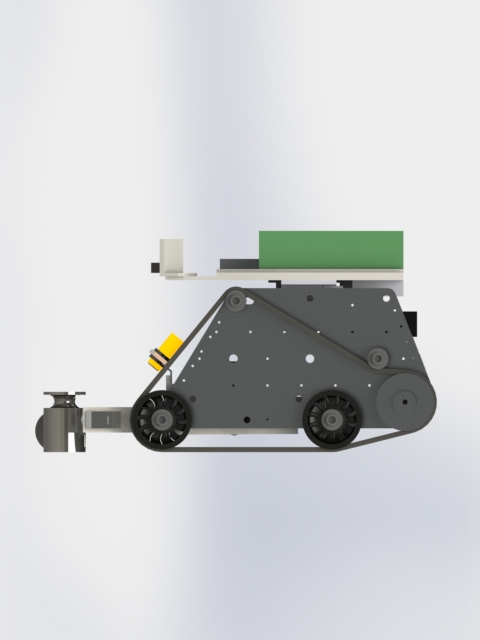
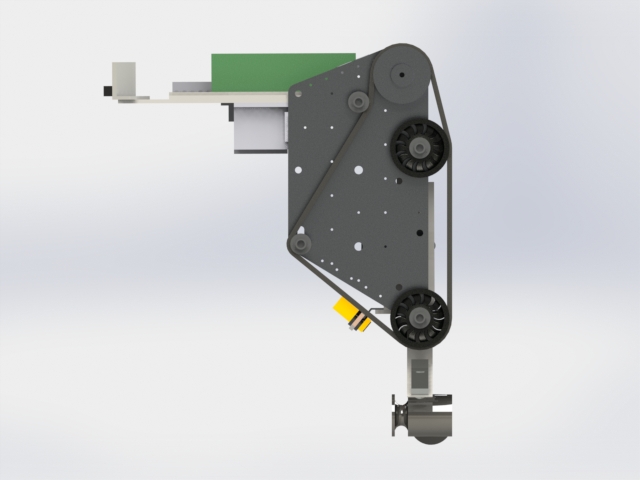
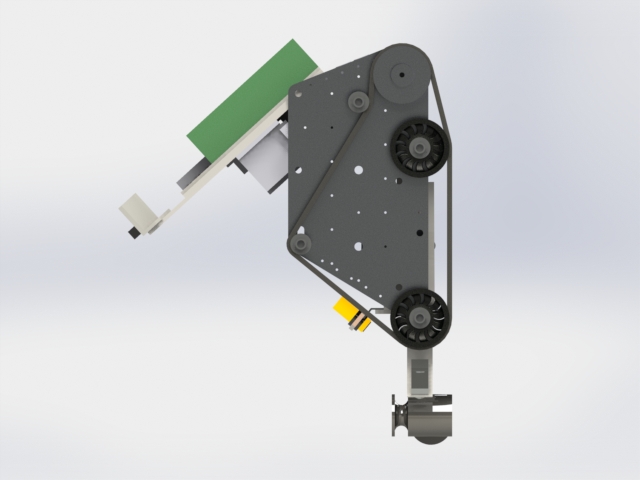


Figure , Shows the quick release mechanism rotate.

As part of the concept specifications, it was required that each group member design a circuit board to aid in the functionality of the robot. These three circuit board designs are presented and evaluated in the following sections.

### Circuit 1: Small speaker amplifier

The circuit is used to amplify the signals outputted from the Arduino as they are too weak to drive the speaker directly with a significant volume. This circuit will consist of a LM386 IC chip, along with the necessary resistors and capacitors to drive a 1 watt speaker at 8 Ohms. The circuit will have a variable volume control via an adjustable pot.

### Circuit 2: Switch bank

There are several values and parts of code which can be enabled and disabled, changing them requires the code to be recompiled and uploaded. To prevent this, Circuit 2 will have a bank of eight switches which can change Boolean values in the code during run-time. Such variables include whether or not to play sounds, send serial or to be controlled manually. See Appendix ### for schematics.

### Circuit 3: IR LED Mount Board

This board will mount X IR LEDs, providing power and ground. A switch will also allow the LEDs to turn off. These LEDs will be used to light up the packages and the IR camera will return the four greatest IR sources. The camera will be mounted in the centre of the board with the LEDs around it.

## 3.2 Sensors

The sensors are a vital component to the correct operation of our robot. The sensors are required to be placed in such a way that will allow them to detect their environment accurately and relay this information to the Arduino. The robot design currently implements the following sensors.

### Obstacle Avoidance

In order to navigate the arena, a combination of two medium and one long range IR sensors were implemented for obstacle collision avoidance. IR sensors are much better suited to detecting obstacles as they have a narrower beam width and do not suffer from beam deflection.

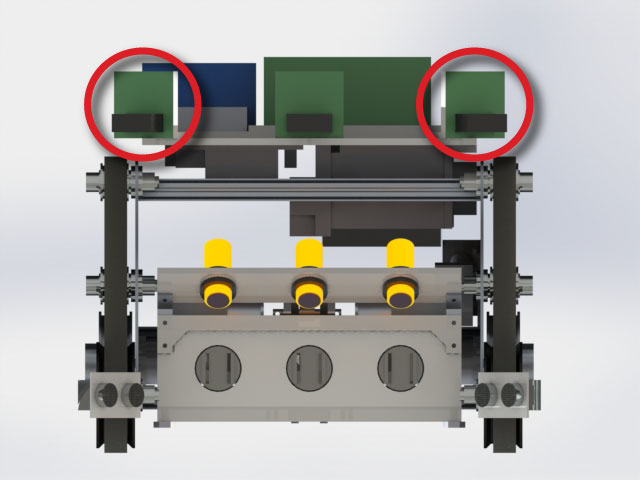
The medium IR sensors were used for obstacle collision avoidance as they could operate efficiently with the range of 100-800mm. These sensors were mounted on the front of the robot and set at an angle of 35 degrees. See figure 9 for sensor position. This would allow the medium IR sensors to work in tandem with the ultrasonic sensors, allowing the robot to distinguish between obstacles and weights.

Figure , Shows the location of the medium range IR sensors.

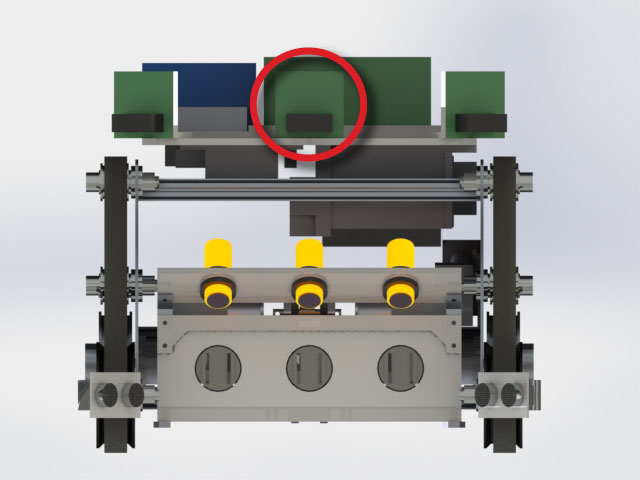
For obstacle collision avoidance with poles, the long IR sensor was used predominantly. It was placed in line with the medium IR sensors at the front of the robot. The IR acts to fill in the blind spot that is created by the medium IR sensors. Figure 10 indicates the location of the long range IR. **This is shown in the following figure, note how the current sensor configuration has significant blind spots that should be addressed.**

Figure , Shows the location of the long range IR sensor.

The IRs were required to be calibrated to allow the robot to know approximately how far obstacles are situated in relation to itself. This was done by measuring the signals from the sensor at varying distances and saving their value to the robot's configuration file.

### Weight detection

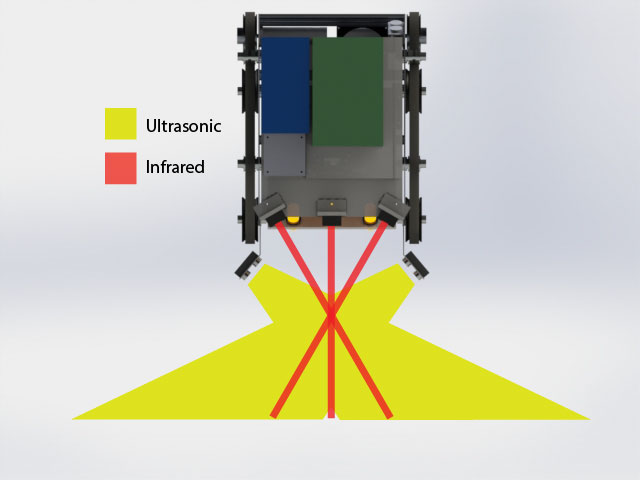
It is evident that the robot must have the ability to detect weights in the arena and bring them on in a permanent fashion. To achieve this, the provided ultrasonic sensors we utilised. Ultrasonic sensors are much better at detecting weights due to their broad beam width and ability to detect cylindrical objects. Two ultrasonic sensors were mounted at the bottom of the robot in front of the tracks. By comparing the signal difference between the top mounted IRs and the bottom mounted ultrasonic, the robot is able to distinguish between obstacles and weights. The ultrasonic sensors were set at an angle of 40 degrees to allow a broad sensor coverage for the front of the robot. The mounting for the sensors are shown in figure 12.

Figure , Shows the birds eye view of the sensor placement.

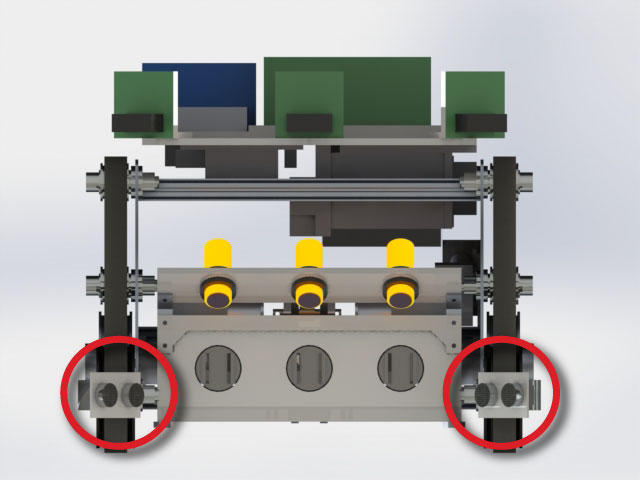
Unlike the IR sensors, the ultrasonic were not required to be calibrated as they operate on a function that takes into account the speed of sound which determines the distance based on the delay.

Figure , Shows the location of the ultrasonic sensors.

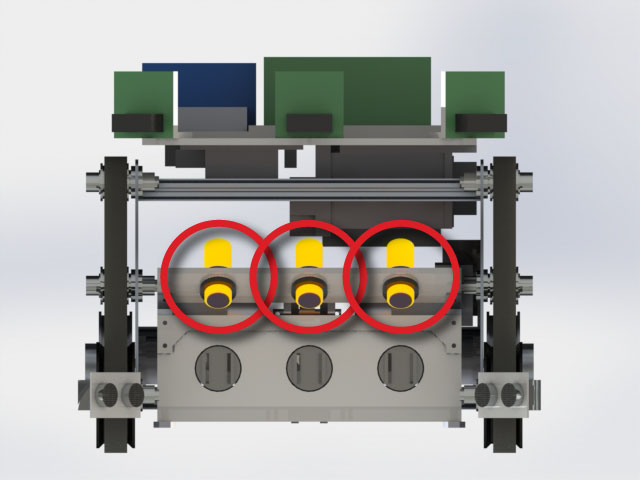
Variable IR sensors were used as on board weight detectors as they had a simple true or false signal return. When weights were picked up by the magnets, the variable IR will trigger, allowing the robot to know the number of weights currently on board. These sensors are situated directly above the permanent magnets on the pickup module. This can be clearly seen in figure 13.

Figure , Shows the location of the variable IR sensors.

# 4.0 SOFTWARE DESIGN

## 4.1 Program Structure

As an autonomous robot, the code is complex and needs to be well structured. Before writing anything, the different modules were created in order to keep everything tidy and fulfil R5.4. The program was implemented primarily from a bottom-up approach, creating the low-level sensor and actuator functions, followed by a top-down approach. This meant the robot did not have to be physically built initially, besides plugging in the components on a temporary basis to test code.

Each module is written to be as independent from others as is reasonable, with a level of abstraction for other modules to reference. Their interfaces with other modules are demonstrated in Figure 14 while the support functions are used by most other modules. While much code for a map and SD card referencing have been written, they were not used in the functional assessment.

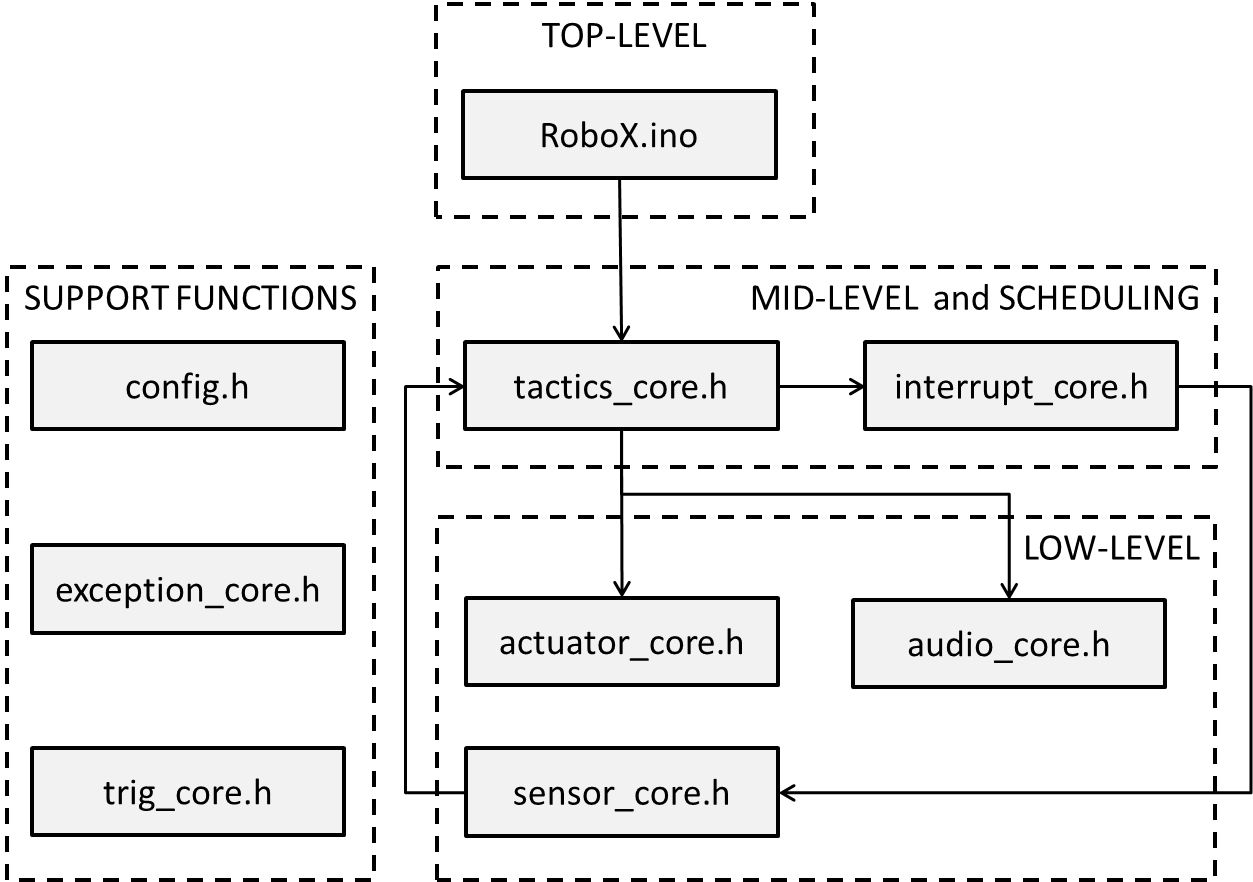


Figure , the files and how they interact

## 4.2 Support Functions

### config.h

To make adjustments to the robot simpler and all in one place, a configuration file was written. It contains no functions or classes, but only macros. This includes the components pin-out, sensor calibration and various program parameters. This file only has values that are subject to change, other macros are also defined in the relevant modules to keep the program clean. Several definitions are there to enable or disable various aspects of the program in order to save processing time. These include whether or not to print serial, play sounds or use the map.

### exception\_core.h

The exception core is a design decision intended to make it easier to code ways around potentially crippling errors. It contains a class which is essentially an error flag with a wrapper. If there is some problem (such as the motors do not initialise), then it raises the flag and prints the error (or plays a sound). Another section of code detects the error and takes a different course of action. This is intended only for functionality issues rather than normal program operation such as detecting a base.

### trig\_core.h

Since the intention of the program is to eventually use mapping and localization, trigonometric functions will be an essential set of tools. This module includes two classes and several macros for radian and degree conversions. The two classes are a two-dimensional Cartesian vector and a polar vector, and use overloaded functions to make it as easy to use as possible.

Since the vectors would added multiple times throughout the code, the efficiency of two different methods were compared. When adding two Cartesian vectors, it is simply a matter of adding components, but when adding two polar vectors, the sum can either be calculated directly or added as polar vectors. The comparison can be found in Appendix 4, which showed that converting to Cartesian coordinates, adding and converting back is more efficient than using trigonometric functions. This led to the decision to use Cartesian form as often as possible to make it even more efficient.

## 4.3 Top-Level Design

Since the robot is autonomous and will have no human intervention (R1.1), the code must be as robust as possible, and have backup systems in case of failure. To accommodate this, there is a loop even higher than the main routine. There are a total of four ‘main loops’ which the program can fall into depending on the state of the robot. Each of these is found in tactics\_core.h.

### Idle Loop

When the robot starts, each module is initialised and the program falls into the ‘idle’ loop. This is simply so that the robot can wait for user input that the round is starting. The robot will not fall back into this loop unless there is human intervention (the round has finished) or everything else has failed.

### Primary Loop

When the round has started, the program begins the Primary Tactic. Under normal circumstances, the robot will perform Simultaneous Localisation And Mapping (SLAM), pathfinding and high level decision-making. If there is any crippling error such as losing the SD card, then it will drop to the Secondary Tactic.

### Secondary Loop

This routine is meant to pick up weights with minimum complexity. This means there is no mapping or planning, but only on-the-spot decision-making. Since there is still a colour sensor, the robot is still capable of returning packages to base and avoiding the opponent’s base, the only disadvantage over the Primary Tactic is that it will be less efficient. It should be noted that this is the code that was developed for the functional assessment so that there is a fall-back for the Primary Tactic.

### Manual Loop

There is a fourth loop which has been implemented, and gives full control to the user. The user can send commands through the serial terminal to make the robot manoeuvre and move other actuators, and the terminal prints the sensor readings. This mode has been used extensively for testing, though will be less useful now that the development is focussed on the other tactics.

## 4.4 Mid-Level Design

Each of the main loops have the same basic structure - when the function (tactic) is called, it initialises relevant variables and sets the correct interrupt. This interrupt will read the sensors as well as performing other foreground tasks at regular intervals. After this, it enters the while loop containing background tasks. The idle loop and manual loop have no further design, as they simply wait for user input, so only the two autonomous loops will be discussed in this section.

### Primary Tactic

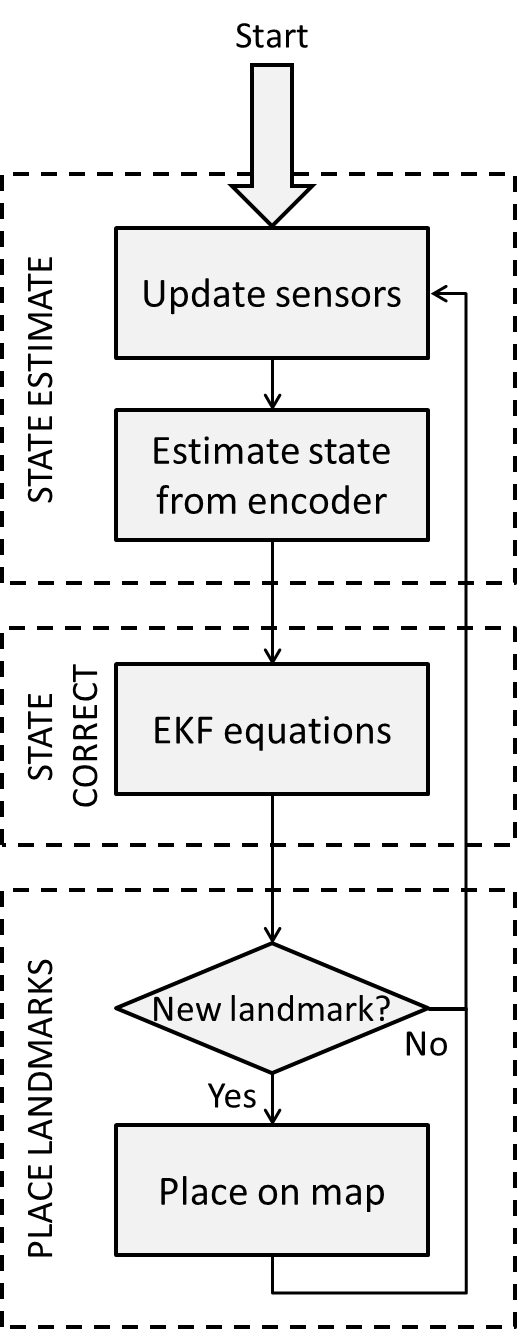
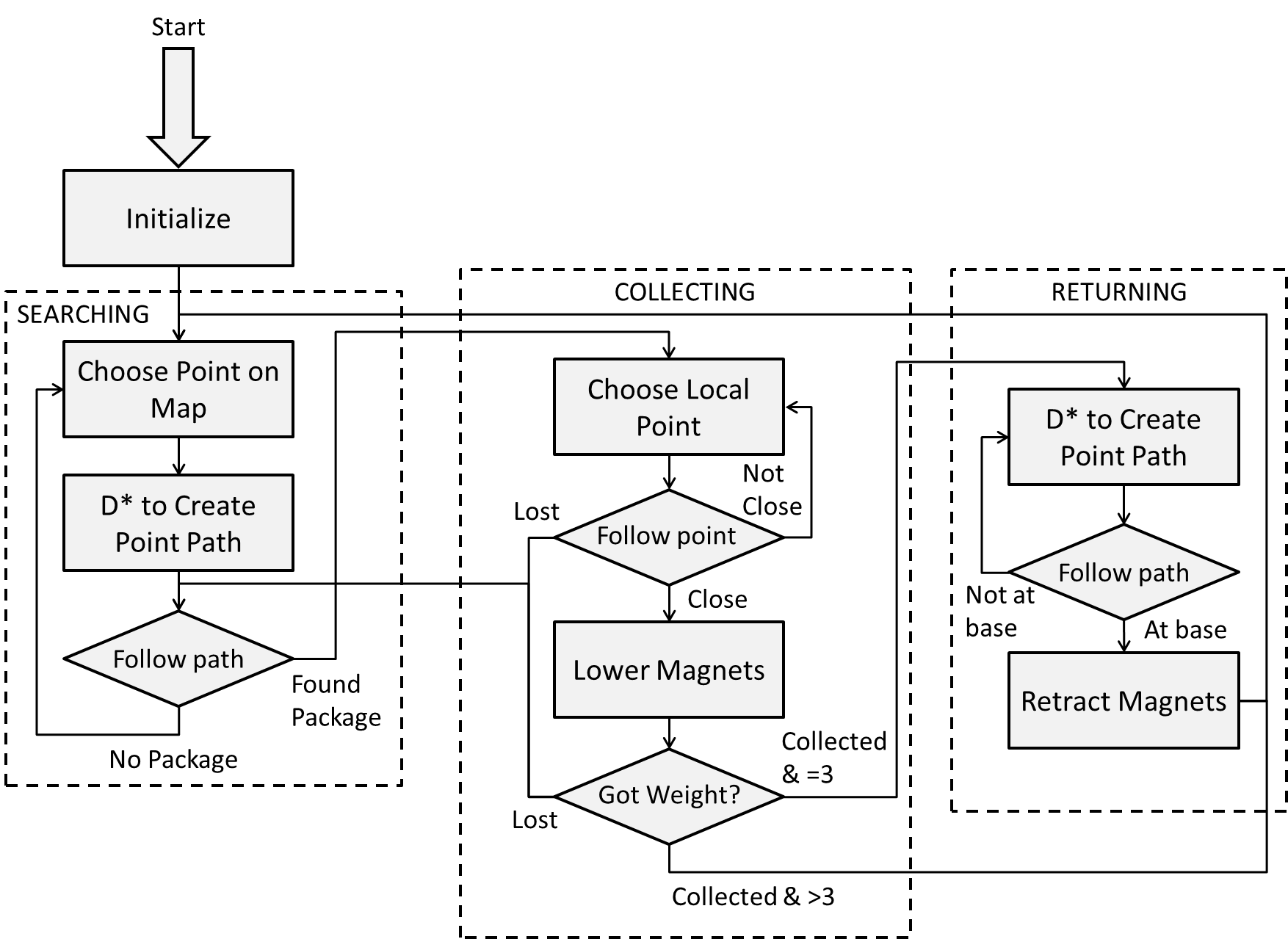
Because the Secondary Tactic was developed for the functional assessment, there are few completed parts of the program. The flow of the program has been written and is presented in Figure ### and Figure ###.

Figure >, Primary Tactic foreground tasks

< Figure , Primary Tactic background tasks



### Secondary Tactic

The overall structure of this tactic is very similar to the primary one since it still intends on completing the same task. The main differences are that the navigation is basically random (whilst avoiding walls), and the robot will drop off weights whenever it can. Unlike in the previous tactic, no processing is done in the foreground, but only updating of the sensors.

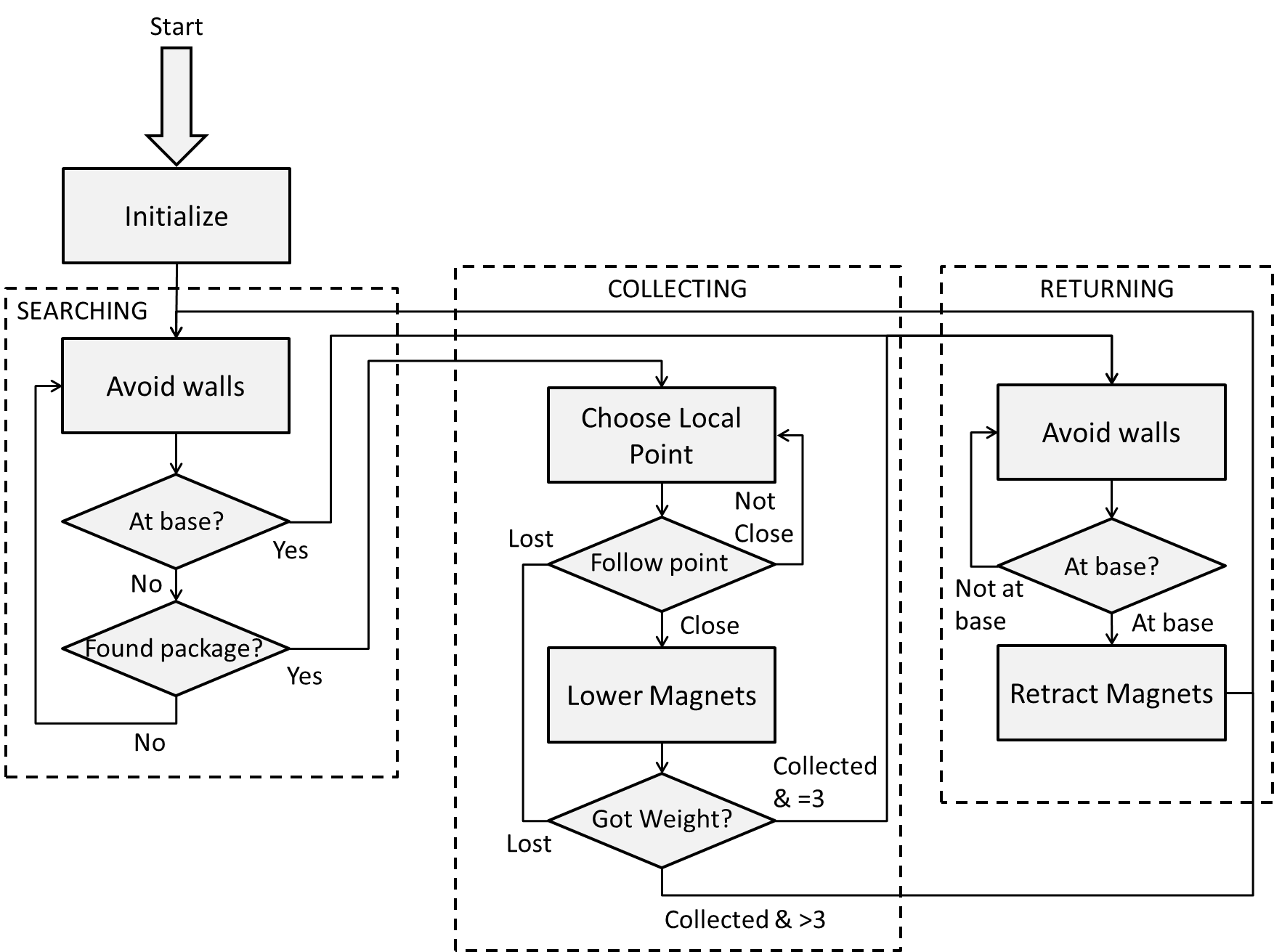


Figure , Secondary Tactic background tasks

## 4.5 Low-Level Design

### Sensors

Although there are many different types of sensors, they were all implemented as classes with the same wrapper. This means that elsewhere in the code, the function calls are the same. Each of the range sensors has five public functions:

* void initialize(parameters)
* void update(void)
* CartVec cart\_read(void)
* PolarVec polar\_read(void)
* bool is\_valid(void)

In the foreground, each of the sensors is updated. This is a straight reading from the sensor with no processing. When this is done, a flag is set to say the value has not yet been processed. When the sensor needs to be read, it processes the value into another register, and decides whether or not the reading is valid based on the sensor type. If called again, the function will not process it a second time, but just returns the value. The reason there are two different types of reading is because they are useful in different scenarios. polar\_read() returns a distance (mm) and an angle, whereas cart\_read() returns the detected object position relative to the centre of the robot (in Cartesian coordinates).

As for the other sensors, they only have three public functions:

* void initialize(parameters)
* void update(void)
* type read(void)

These are the same as for the range sensors, only they return various different values depending on their function.

### Actuators

As with the sensors, each different actuator (DC motors, servos, stepper motors) were implemented as classes. Since they each have different functions, they are not referenced exactly the same. The stepper motors are given a relative position, servos are given an absolute position, and the DC motors are given a scaled speed and rotation.

# 5.0 Design Evaluation

## 5.1 General

#### R1.1 The robot will be fully autonomous ✔

#### R1.2 The robot will be controlled by the Arduino Mega ADK supplied ✔

#### R1.3 The robot shall be able to move, identify and collect packages ✔

#### R1.4 The robot shall operate until all 11 packages are claimed or the time limit is reached ✘

Currently, the robot can act fully autonomous with the provided Arduino Mega ADK. It can move around the arena and collect weights consistently. It is yet to be tested with collection of more than three packages as this requires returning to base to drop off the collected weights.

## 5.2 Identification

#### R2.1 The robot shall be able to identify food packages ✔

To test this functionality, an experiment was setup to observe the detection of packages within the arena. The packages were placed at 0, 30, and 60 degree angles on both sides of the centre line of the robot at a distance of 600 mm. The time taken to detect and retrieve the packages was measured, and it took an average time of 3.93 seconds when it was detected. It could not detect the packages at 60 degrees, but it found 90% of the other packages successfully. Results can be found in Appendix 3.

Based on the findings from this experiment, we can see that weights are easily detected and collected when they are at relatively small angles from the robot’s centre line. Beyond this the ultrasonic sensors are limited by their beam width and set angle. The robot will eventually detect and collect the weights in the arena at any position from the centreline, but it requires the robot to approach from a shallower angle. It should also be noted that the robot produced false detections during the experiment when driving close to the arena walls. It is suspected that the skirting around the arena gives false positives but this needs to be investigated further.

#### R2.2 The robot shall be able to identify obstacles it cannot move over ✔

This was implemented using the medium and long range IR sensors as discussed in the previous report. As a result, the robot performed adequately during the functionality assessment. Its ability to detect poles still lacks accuracy and consistency and should be significantly improved before the final assessment.

#### R2.3 The robot shall be able to distinguish home HQ and the opposition HQ ✘

The colour sensor example code functioned at an acceptable standard, but would not operate in conjunction with the rest of the program. This has not yet been solved, so the robot cannot distinguish between bases. It should be noted that the robot can distinguish the bases as a side-effect of the digital IR sensors – when on a base, all three sensors ‘detect a weight’ due to the change in specularity.

#### R2.4 The robot should rely on a range of navigational sensors ✔

As discussed previously, the robot uses four different types of sensors based on their varying specifications. The range of sensors used is expected to increase by about two times.

## 5.3 Movement

#### R3.1 The robot shall be able to move over obstacles at least 25mm in height ✔

The 3D printed wheels discussed in Section 2 give the robot a ground clearance of over 25mm. They were designed this way and measured using Solidworks.

#### R3.2 The robot shall be able to fit through gaps of at least 500mm in width ✔

When placing the robot in between two walls, it successfully drives though the gap.

#### R3.3 The robot shall be able to manoeuvre around obstacles it cannot move over ✘

The robot can manoeuvre around and follow walls. Unfortunately due to the false detection of weights, it occasionally turns towards and collides with the walls. The robot, while it can detect cylindrical object, currently does not attempt to avoid them.

#### R3.4 The robot shall not leave the designated arena during the competition ✔

It is impossible for the robot to climb the walls.

#### R3.5 The robot should not get stuck in any algorithmic loops for longer than 1 minute ✘

No watchdog timer or other method has yet been implemented to avoid this. When mapping and D\* are functional, it may not even be necessary to use a timer.

## 5.4 Collection

#### R4.1 The robot shall be able to pick up a package so that it is under the robot’s control ✔

At this point the design can collect packages fast and effectively, as confirmed in the functionality assessment, and discussed under R2.1.

#### R4.2 The robot shall have a way of carrying, at most, three packages without hindrance ✘

Three packages are able to be picked without trouble. The tilt operation has not yet been implemented, so it is unable to prevent the packages from interfering with level obstacles.

#### R4.3 The robot shall not collect any packages within the opposition’s HQ ✘

The tilt operation has not been implemented and thus will collect packages in the opposition’s HQ.

#### R4.4 The robot should be able to release any packages it has on-board to HQ ✘

Much work was put into a release mechanism, as discussed in detail previously, but is unable to release more than one package. Minor developments and fine tuning is required to get enough force on the magnets.

#### R4.5 The robot should be able to pick up packages in any orientation and any part of the map ✘

The magnets attach the packages in any orientation, but is unable to pick up packages against a wall.

## 5.5 Construction

#### R5.1 The cost of additional items shall not exceed $50 (except for R5.2) ✔

The additional items as priced in Appendix 5 sum to $##.##.

#### R5.2 Each member shall design their own PCB for use on the robot, not exceeding $5 ✘

Blah

#### R5.3 The robot should be built with less than 200g of 3D printer plastic. ✔

The four 3D printed wheels, each printed in two halves weighed a total of 152 grams.

#### R5.4 The robot shall be easy to maintain and disassemble ✔

By making the all the circuit boards mounted on a single sheet of perspex, it allowed for ease of total removal from the robot by the removal of the three mounting screws. For simple rewiring, the quick release with its swivel design could allow access to all the circuit boards in 12 seconds.

## 5.6 Safety

#### R6.1 The robot shall not cause any deliberate damage to anything or anyone ✔ (has drawn blood)

#### R6.2 The robot shall have an accessible ‘off’ switch ✔

#### R6.3 The robot shall use the battery safety circuit provided ✔

For the electrical module of the robot design, it was necessary for the robot to meet several design specifications., an accessible ‘off’ switch and the use of a battery safety circuit (R6.2, R6.3). The power switch was mounted at the back of the robot so it could easily be access. It will be necessary in the final assessment to relocate the switch as it could be disabled by the opposition robot.

# 6.0 Future Development

## 6.1 Mechanical Development

The pickup mechanism needs fine tuning to make sure all the force is transferred to the magnets. This will be done by reinforcing the structure of the mounts and by connecting all the magnets together.

For a greater force on the magnets two rubber bands may be added. This will give more force on the back the stroke of the mechanism where it needs the greatest force. When extending the magnets back out the mechanism will put energy into the rubber bands.

Having the pickup mechanism rotate up and down design also has to be finalised. At this stage two servos will be mounted above to raise and lower it.

Keeping the tracks on the wheels is also needs further development. An extra bearing will be added to support the track on the bottom of the robot. Consideration are also being taken for adding a spring to the bearing to account for when that section of the track goes over a bump.

A switch is need to tell when the stepper motors have rotated around past the point of knocking of the packages. This will ensure the packages have indeed disconnected with the robot.

## 6.2 Electrical Development

Currently based on the sensor and circuit design, the robot functions on a basic level. It has the ability to navigate most obstacles and detect weights the majority of the time. It is now necessary to improve the reliability of weight detection and obstacle detection. There are a number of potential design paths that we are currently considering for environment detection and they are discussed in brief below.

### IR camera

The supplied IR camera has the ability to detect four IR sources and give you the exact x and y coordinates. If IR LEDs were used in conjunction with this camera, it could be used to detect weights within the arena. The weights would reflect the IR waves and appear as the brightest spot on the camera as it would reflect more than the surrounding darker obstacles. It does have some drawbacks including the risk of interference from the opposition robot. This would theoretically provide far better identification and tracking for weigh detection in the arena, but it has been untested as of yet.

### Camera

Another option for weight and obstacle detection is the use of the supplied camera circuit board. With the resolution of 640 x 480 pixels, the camera would easily be able to distinguish between objects in the arena. The downside to this camera would be the time taken to write and debug the program, not to mention the processing time required to obtain images in real time. This development will only be investigated if time permits.

### Swivel mounted IR

If both the IR and Arduino camera proved unsuccessful, it would be possible for two IRs to be mounted on small servo motors. These motors could be driven back and forth to produce a sweeping motion for the IR sensors. An array of values could be produced from this, allowing objects such as weights and obstacles to be detected. This method could potentially be more processor intensive but more testing is required if this option is to be ruled out.

### Side mounted IR

Side mounting IR sensors will help with the navigation of the robot around its environment as the robot would have a better idea on its current location within the arena. This will also be beneficial for when and if the pathfinding is implemented on the final robot design.

### Sonar

A central mounted Sonar sensor instead of a long range IR that is currently utilised on the robot would be better and detecting poles as it produces a large beam width similar to the ultrasonic sensors. However, they are generally less reliable that ultrasonic or Infra-red sensors. They are better and determining if something is detected rather than how far away it is.

### IMU

The IMU contains many useful navigational modules that can be utilised to aid in the robot's path finding capabilities. Accelerometer and gyro modules are just some of the options utilised in this board. The magnetometer may not be ideal seeing as the pickup mechanism relies on high power permanent magnets. Discuss further with Peter :P

### Colour Sensor

For the robot to detect the arena bases, the colour sensor must be implemented in the design. This sensor should be placed as near to the front of the robot as possible to allow for ease of detection. The exact location of this sensor is yet to be determined.

## 6.3 Software Development

Stuff

# Contribution Statements

### JACK

* Report electrical section
* Proof reading
* Design and build of electrical module

### PETER

* Software design and implementation
* Report software design section
* Report formatting
* Circuit 2 design
* Part sourcing

### RYAN

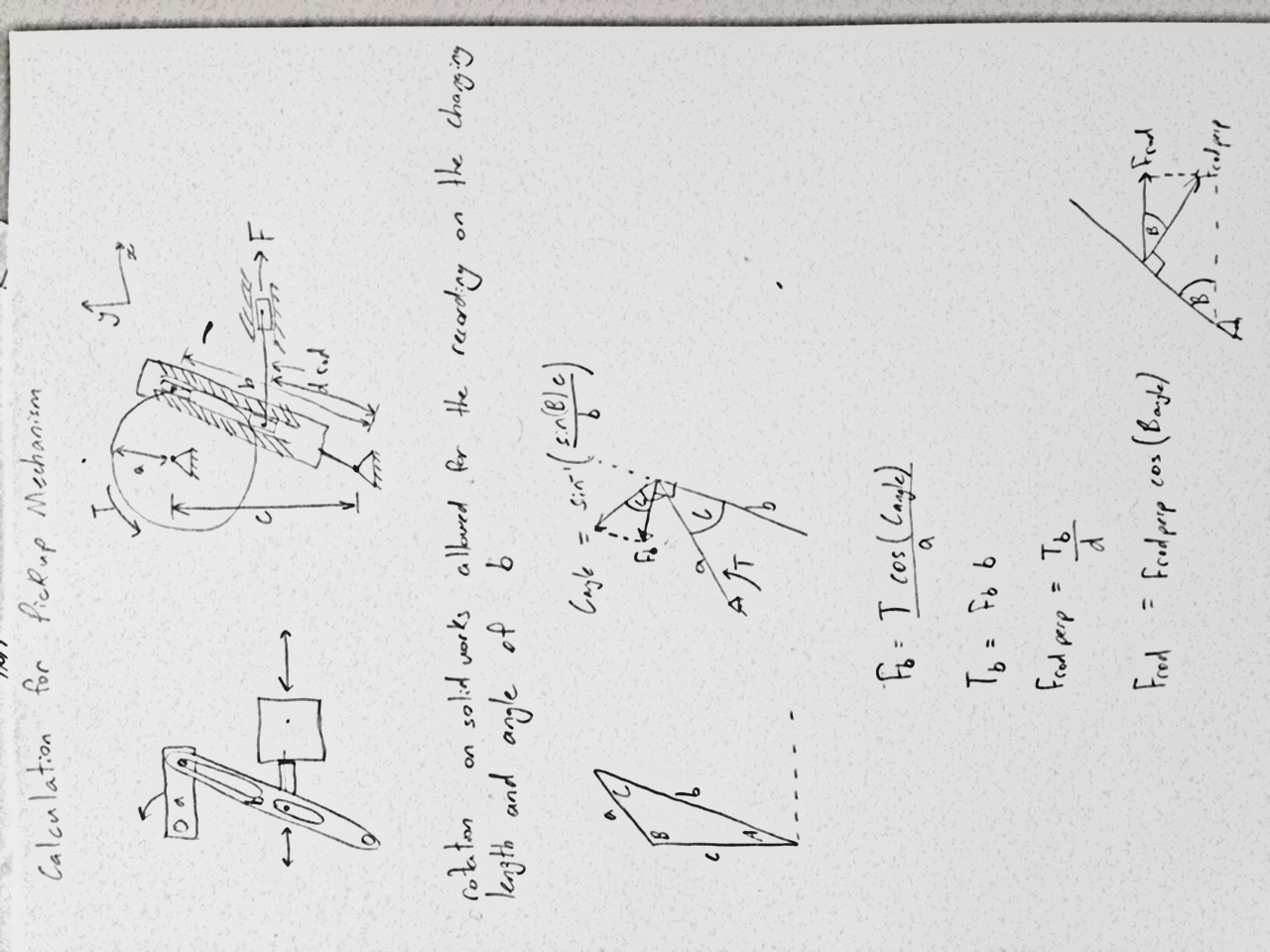
* Report mechanical design section
* Report formatting
* CAD modelling/drawing
* Pickup mechanism design, calculations, Matlab coding and build/machine (85 hours )
* Material/Parts sourcing
* Circuit 3 design

16/25

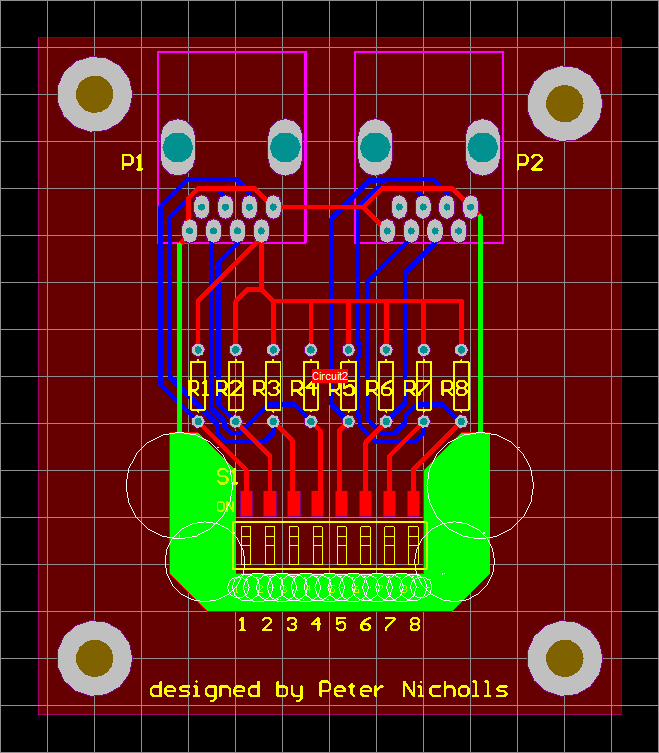
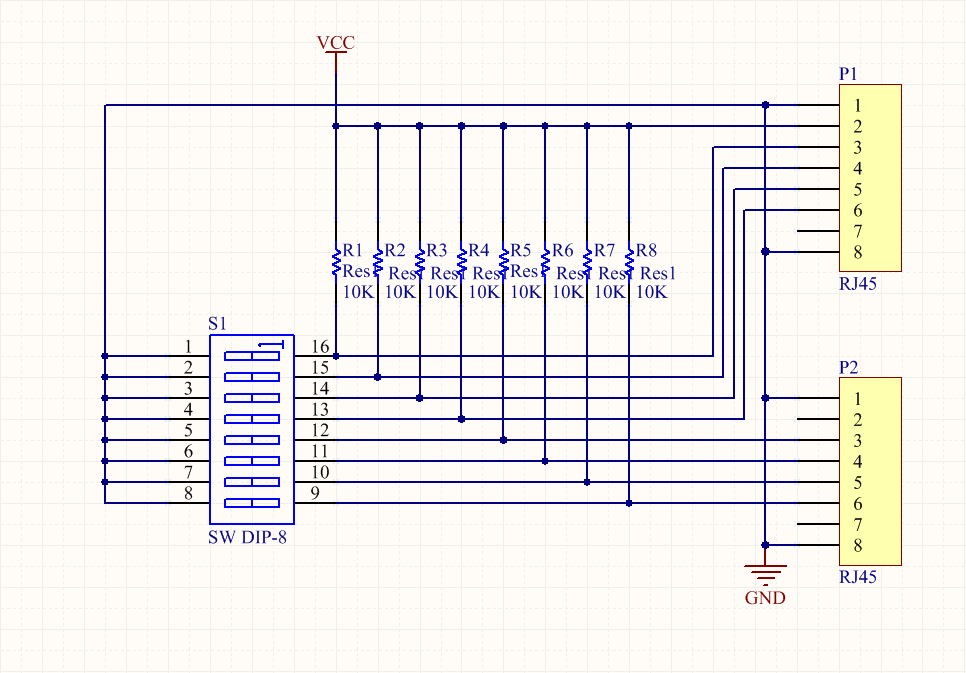
9/25

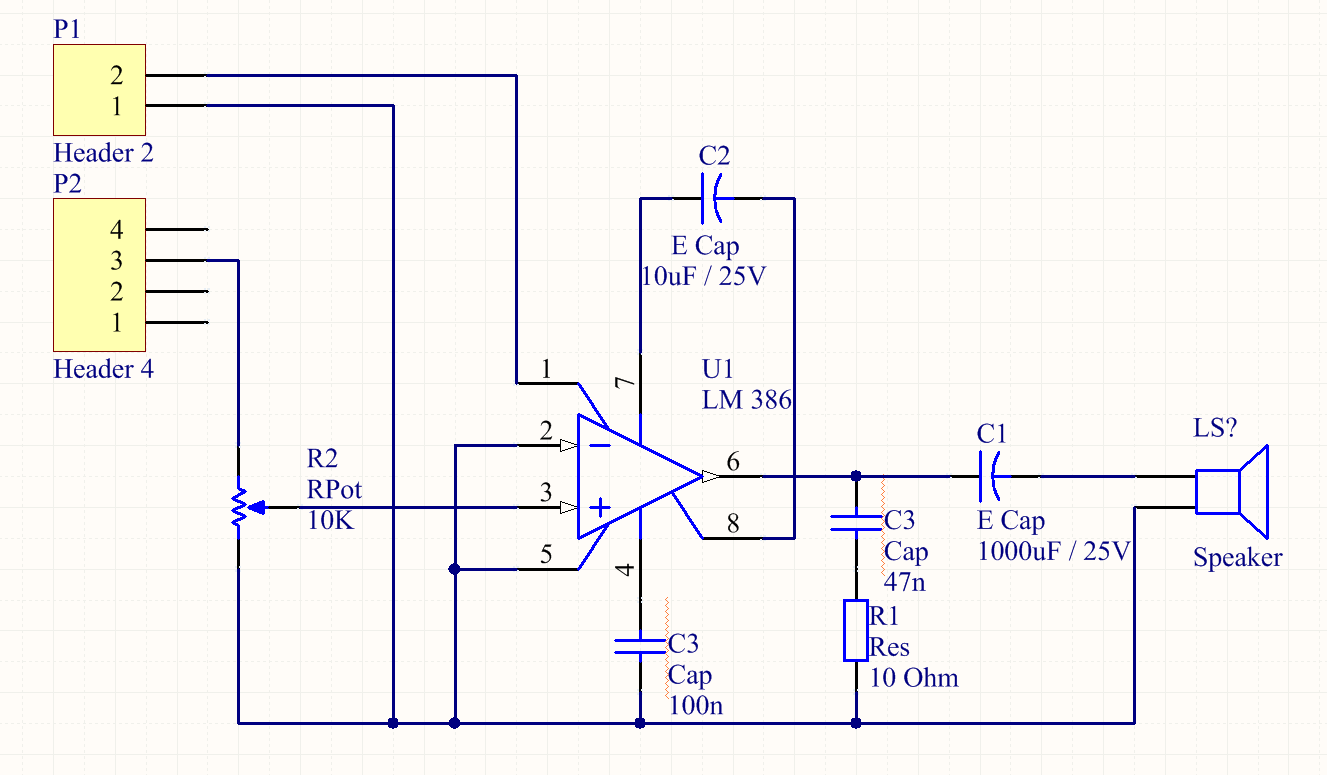
# Appendix 1 – Matlab Code and Derivation

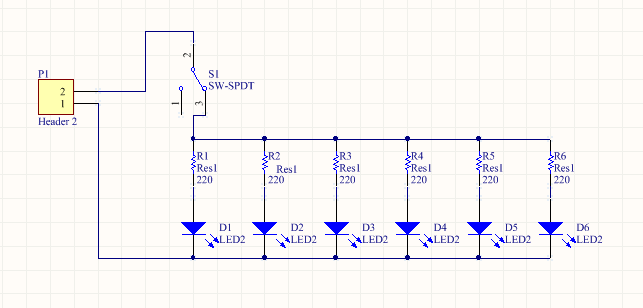
|  |
| --- |
| clear, clc  close all    force\_required = 150; %N  motor\_torque = 1.76\*2; %Nm    radius\_gear = 0.010; %needs to be more then half the distance of our tavel  radius\_rod = 0.030;  off\_set = 0.050;  % 0 10 20 30 40 50 60 70 80 90 100 110 120  b\_length = [61.3 63.2 64.8 66.1 67.4 68.6 69.4 70.5 70.6 70.6 70.5 69.4 68.6 67.4 66.1 64.8 63.2 61.2 60 58 56.4 54.7 53.6 52.3 51.4 51 50.7 51 52.3 53.6 54.7 56.4 58 60 60.8 61.3];  b\_angle = [9.5 9 8.4 7.6 6.6 5.3 4.0 2.7 1.5 0 2.7 4.0 5.3 6.6 7.6 8.4 9 9.5 9.6 9.4 8.9 8.1 7.2 5.6 4 2.2 0 2.2 4 5.6 7.2 8.1 8.7 9 9.2 9.5];  a = 1;    for i = 1:1:36 ;  angle\_c(i) = asind(sind(i\*10+90)\*0.05/(b\_length(i)/1000));  end  for i = 1:1:36  force\_b(i) = motor\_torque\*cosd((angle\_c(i)))/0.01;  end  for i = 1:1:36  torque\_b(i) = force\_b(i)\*b\_length(i)/1000;  end  for i = 1:1:36  force\_rod\_per(i) = torque\_b(i)/0.03;  end  for i = 1:1:36  force\_rod(i) = force\_rod\_per(i)\*cosd(b\_angle(i));  end  for i = 1:1:36  force\_needed(i) = 150\*3;  end    plot(force\_rod,'r')  grid on, hold on  plot(force\_needed)  axis([0 37 0 900])  title('Force applied to magnets')  xlabel('Motor angular displacement')  ylabel('Force applied to magents')  legend('Force applied','Force needed')    motor\_force = motor\_torque / radius\_gear  applied\_force = motor\_force \* (off\_set / radius\_rod) |

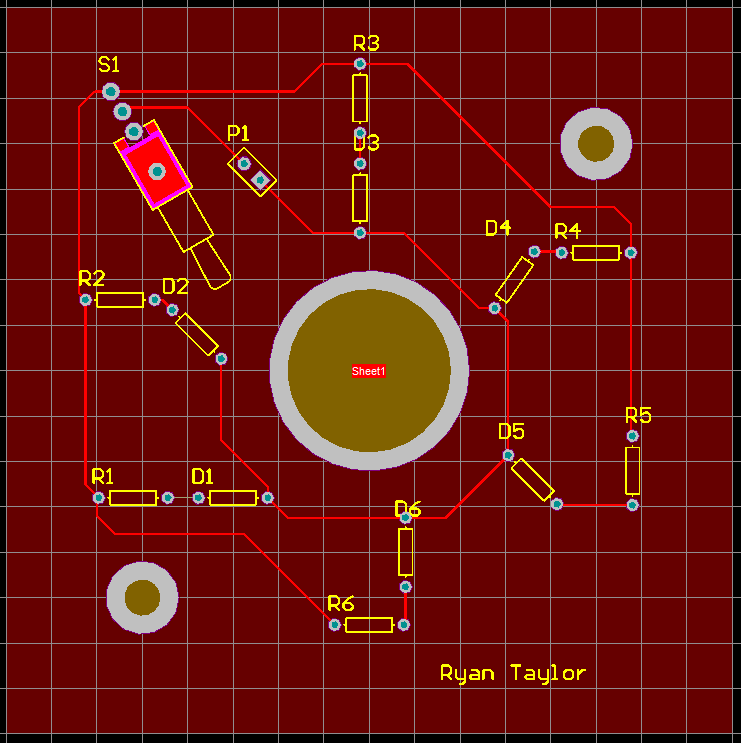
****

# Appendix 2 - Circuits

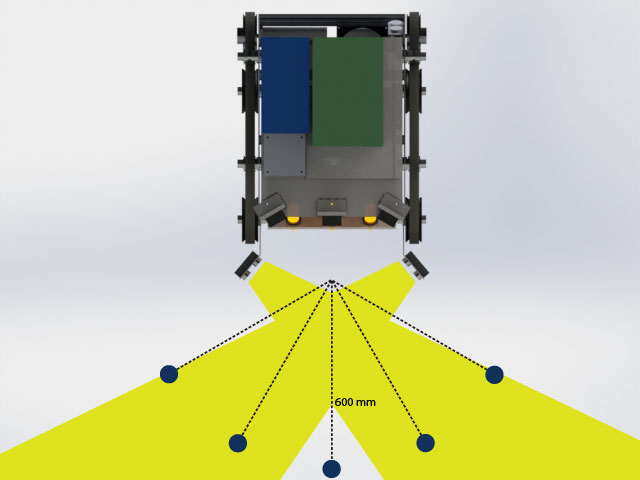






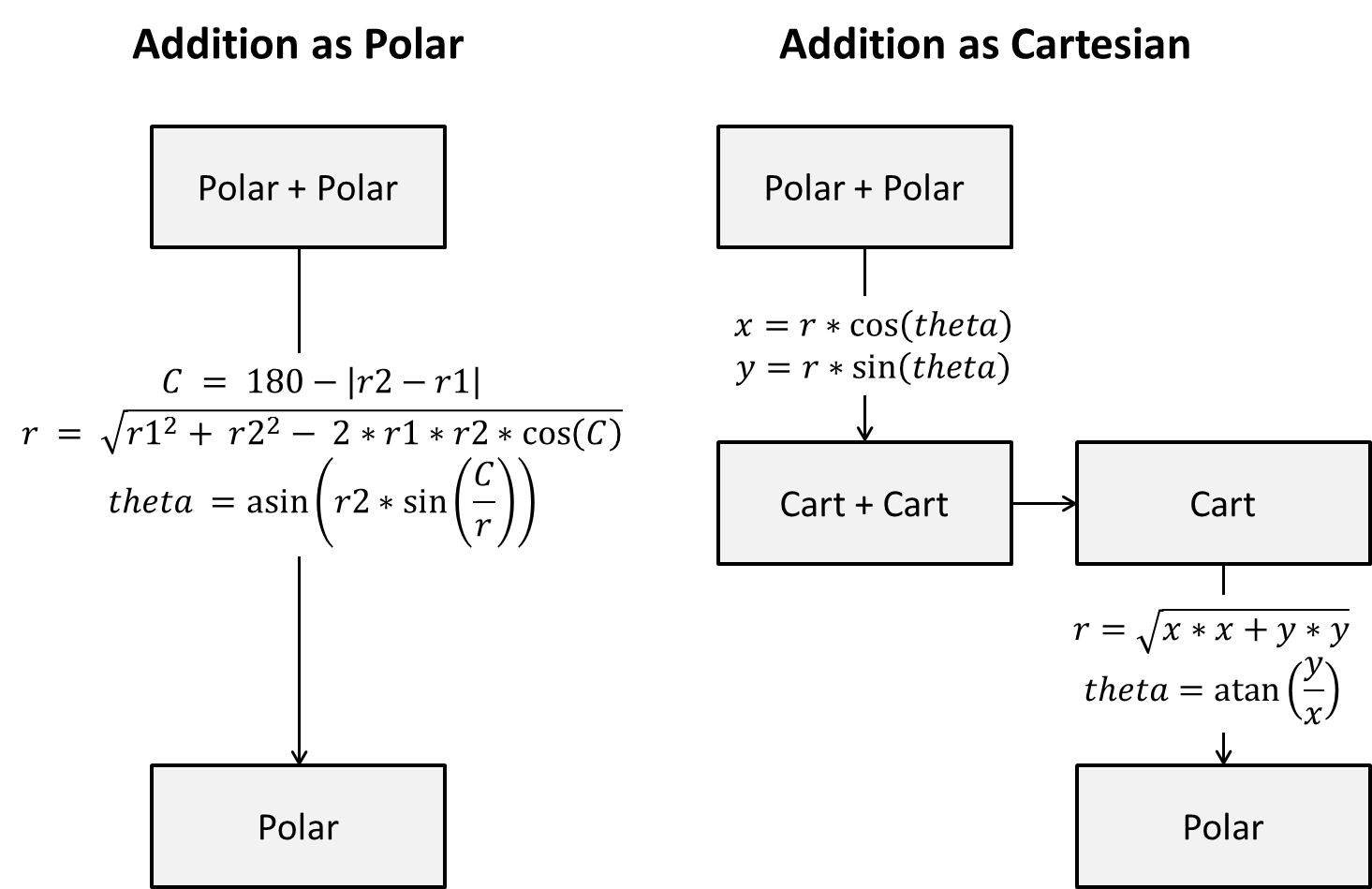


# Appendix 3 – Package Detection



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Centre line | 30 deg anti | 60 deg anti | 30 deg clock | 30 deg clock |
| 1 | 3.88 | 3.80 | Not detected | NA | Not detected |
| 2 | 4.00 | 3.96 | Not detected | NA | Not detected |
| 3 | 3.90 | 3.86 | Not detected | 3.96 | Not detected |
| 4 | 3.95 | 3.93 | Not detected | NA | Not detected |
| 5 | 3.90 | 3.82 | Not detected | 4.08 | Not detected |
| 6 | 3.92 | 3.80 | Not detected | 3.91 | Not detected |
| 7 | 4.04 | 3.88 | Not detected | 3.93 | Not detected |
| 8 | 4.14 | 3.84 | Not detected | 3.80 | Not detected |
| 9 | 3.91 | 3.93 | Not detected | 4.08 | Not detected |
| 10 | 3.64 | 4.13 | Not detected | 4.00 | Not detected |
| Average | 3.93 | 3.90 | Not detected | 3.97 | Not detected |

# Appendix 4 – Efficiency Tests



Since the majority of vector addition will be adding an offset to the sensor, the offset can be saved in its Cartesian form. This would imply one fewer conversions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Polar | Cartesian | Difference | Efficient Cartesian | Difference |
| Add, Subtract | 4 | 2 | +2 | 2 | +2 |
| Multiplication | 8 | 6 | +2 | 5 | +3 |
| Division | 1 | 1 | 0 | 1 | 0 |
| Absolute Value | 1 | 0 | +1 | 0 | +1 |
| Tan Inverse | 0 | 1 | -1 | 1 | -1 |
| Sine | 1 | 2 | -1 | 1 | 0 |
| Sine Inverse | 1 | 0 | +1 | 0 | +1 |
| Cosine | 1 | 2 | -1 | 1 | 0 |
| If Statement | 1 | 1 | 0 | 1 | 0 |
| Square Root | 1 | 1 | 0 | 1 | 0 |

# Appendix 5 – Bill Of Materials

## Bill Of Materials (Robot)

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Cost (NZ dollar) |
| SD Storage Card | 1 | 4.09 |
| SD Reader | 1 | 2.11 |
| Aluminium plate | 2 |  |
| Aluminium extrusion | 4 |  |
| Sensor Circuits |  |  |
| Arduino Mega ADK | 1 |  |
| Lithium Battery | 1 |  |
| Ultrasonic Sensor | 2 |  |
| Medium Infrared Sensor | 2 |  |
| Long Infrared Sensors | 1 |  |
| Cables | Lots |  |
| 3D Printed Wheel | 4 |  |
| Smart Servo | 1 |  |
| DC motor | 2 |  |
| Drive Wheel | 2 |  |
| Bearing (29mm) | 16 |  |
| M3 Cap Screws | Lots |  |
| M8 Cap Screws | A few |  |
| Total Cost | |  |

## Bill Of Materials (Pick Up Mechanism)

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Cost (NZ dollar) |
| Aluminium 15mm 40x100 | NA | 5 |
| Aluminium 5mm 120x100 | NA | 10 |
| Aluminium 2mm 300x200 | NA | 5 |
| Steel rod 1/4 inch 200mm |  | 1 |
| Steel rod 10mm 100mm |  | 0.5 |
| Spring 20mm compressive |  | 0.1 |
| Perspex 10mm 200x350 |  |  |
| Perspex 4mm 200x150 |  |  |
| M3 | 40 | 4 |
| M5 | 4 | 0.75 |
| Geared Stepper | 2 | 30 |
| Digital IR IO | 3 | 15 |
| Permanent Magnets | 3 | 10 |
| Total Cost | |  |

## Bill Of Materials (Circuit 1)

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Cost (NZ dollar) |
| 1000 uF/ 25V electro cap | 1 | 0.22 |
| 10 uF/ 25V electro cap | 1 | 0.04 |
| 100 n cap | 1 | 0.05 |
| 47 n cap | 1 | 0.40 |
| IC LM386 | 1 | 0.08 |
| 10k variable resistor | 1 | 0.33 |
| 10 Ω resistor | 1 | 0.02 |
| 1 W speaker | 1 | 0.51 |
| Total Cost | | **$1.65** |

## Bill Of Materials (Circuit 2)

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Cost (NZ dollar) |
| SW DIP-8 | 1 |  |
| RJ45 8P8C jack | 2 |  |
| Resistor 10K | 8 | 0.02 |
| Total Cost | |  |

## Bill Of Materials (Circuit 3)

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Cost (NZ dollar) |
| IR LED | 6 | 3 |
| 220 Ω resister | 6 | 1 |
| Header pin | 2 | 0.1 |
| Switch | 1 | 4 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Total Cost | |  |