

Robocup Progress Report 2

Group 8

Matthew Young

Hamish Black

Chris MacEwan

1 Executive Summary

This document, the Detailed Design Report (DDR), details the progress of Group 8 in the 2014 Robocup. The goal of the Robocup is to develop a robot capable of autonomously navigating an arena to find and collect packages of varying mass. Robots compete with a robot from an opposing team to collect the most mass in 5 minutes. In the Conceptual Design Report (CDR) three potential designs for a robot were created and evaluated against a set of project constraints and specifications. A design based on a mechanically operated magnet was chosen as the ideal design and developed into a working prototype.

The chosen design consists of 4 main mechanisms which fulfil dedicated tasks. At the most basic level, the robot must be capable of identifying, collecting and storing packages whilst navigating obstacles. The collection mechanism consists of the magnet, servos and storage bay which collect and store packages. The navigation mechanism is a sensor suite composed of a variety of IR distance sensors, a photodiode array and an ArduCam. These sensors all work in tandem to detect walls, obstacles and packages. Specifically, the ArduCam is used to detect the direction of obstacles and packages, while the IR sensors detect the distance. The photodiode array is used to precisely guide the robot towards packages at short ranges. The locomotion mechanism is entirely comprised of the tracked chassis, upon which every other component of the robot is mounted. The chassis forms the base of the robot and allows it to move around the arena as well as navigate low obstacles. The final mechanism is encompassed by the Arduino and all the software loaded on it. The software coordinates all the various components of the robot, processing data from the navigation mechanism and using it to guide the robot's movement as well as actuate the collection mechanism when a package is within reach.

As part of on-going development, the robot has been tested against the criteria from the CDR to ensure it is operating according to project specifications. As part of this, a functionality assessment was held by the project directors to determine if each robot was able to achieve a set of basic tasks without any performance constraints. I.e. the robot was required to prove that it could move around, identify a package, collect it, and avoid a wall. This was regardless of whether it could do these tasks successfully on every attempt. The functionality assessment also required the robot to perform these tasks autonomously. Group 8 passed every test in the functionality assessment.

In addition to the functionality assessment, the robot has been assessed by the designers against some of the performance specifications from the CDR, to ensure that the robot is achieving tasks with sufficient speed and precision to allow it to beat opposing robots in the Robocup. The robot does not yet have full functionality, so only some of the requirements have been tested. The robot has specifically been tested to ensure that it can:

- Lift the heaviest package reliably
- Store 7 packages on-board
- Collect a package within a time limit
- Manoeuvre with a variety of payloads
- Collect packages against a wall
- Collect packages in corners

The robot can accomplish all these tasks successfully with the exception of traversing obstacles with a 5 kg or greater payload.

Based on the requirements that have not been met or are yet to be met, areas for further development have been discussed at the end of this document.

Contents

1	Executive Summary	3
2	Introduction.....	6
3	Design Description.....	7
3.1	Method of Operation	7
3.2	Collection Mechanism	7
3.2.1	Package acquisition.....	7
3.2.2	Package storage	9
3.3	Navigation Mechanism.....	9
3.3.1	ArduCam.....	10
3.3.2	Infrared distance sensors	12
3.3.3	Infrared photodiode array.....	13
3.4	Locomotion Mechanism	14
3.5	Software Architecture.....	16
3.5.1	The Main Thread	16
3.5.2	Drivers	16
3.5.3	Services.....	17
4	Evaluation	17
4.1	Results of the Functionality Assessment	17
4.2	Collection Mechanism	18
4.3	Navigation Mechanism.....	19
4.4	Locomotion Mechanism	19
4.5	Software.....	22
5	Further Development.....	23
5.1	Current Requirements	23
5.2	Additional Developments	24
6	Contribution Statement	25
6.1	Matthew Young.....	25
6.2	Hamish Black.....	25
6.3	Chris MacEwan	25
7	Appendices	26
7.1	Appendix A: Bill of Materials	26
7.2	Appendix B: Load Calculations for Collection Mechanism	27

7.3	Appendix C: Raw Data for Package Collection Timing.....	28
7.4	Appendix D: Raw Data for Robot Mobility Under Load.....	28
7.5	Appendix E: Storage plate schematic.....	30
7.6	Appendix F: Chassis extension manufacturing drawings.....	31
7.7	Appendix G: Image Processing Test Sets.....	33
7.8	Appendix H: Image Processing MATLAB Script.....	34
7.9	Appendix I: Photodiode Array Driver Circuit.....	35
7.10	Appendix J: Magnet Housing Schematic.....	36

Table of Figures

Figure 3.1: CAD model of package acquisition sub-system and its exploded view.....	8
Figure 3.2 Example of package collection.....	8
Figure 3.3: Flow Diagram of Magnet Operation.....	8
Figure 3.4 Storage plate and storage plate in cutaway of chassis assembly	9
Figure 3.5: Image processing steps to detect red obstacles	10
Figure 3.6: Unprocessed test image.....	11
Figure 3.7: An image generated by applying the score function to the original test image.....	11
Figure 3.8: IR distance sensor cone of vision	13
Figure 3.9: Three possible package placement scenarios in front of the IR array.....	14
Figure 3.10: Revised chassis design.....	15
Figure 3.11: Robot Software Architecture.	16
Figure 4.1: Time to traverse a 1 m obstacle	21
Figure 4.2 Collection of package located in corner.....	21
Figure 5.1 Upright orientation, first sideways orientation, second sideways orientation	24

Table of Tables

Table 4.1: Functionality assessment results.....	17
Table 4.2: Time to traverse 2 m.....	20
Table 4.3: Time to rotate 360 degrees.....	20
Table 7.1: Time taken to collect a package in seconds.....	28
Table 7.2: Time taken to traverse 1 m	28
Table 7.3: Time to traverse a 1 m obstacle.....	29
Table 7.4: Time to rotate 360 degrees.....	29

2 Introduction

Every year, 2nd Pro Mechatronics students at the University of Canterbury compete in the Canterbury RoboCup Competition. Students are split into teams of three, trying to design a robot to be the very best, like no one ever was. This year the scenario is search and rescue in the wake of a zombie apocalypse. Two teams battle head to head within a “city” scattered with “food packages” over the course of 5 minutes. To catch each “food package” represented by metal cylinders of varying mass between 0.5 and 1 kg is the robot’s real test, to store them on board the robot or at their respective home base is its cause. The robot must be able to travel across the arena, searching far and wide for the packages using a variety of sensors. Once all packages are collected or the time limit is up, the robot that has collected the most cumulative mass wins. Each team must face off each other in a round-robin style competition, with the winners facing the winners of other rounds and the losers facing the losers of the other rounds as well. At the end of the day every team will have been “sorted” from best to worst.

The “city” is an arena 2.4 x 4.9m in size with 400mm high walls surrounding it. The arena contains a number of obstacles scattered throughout with the final location of each not being revealed until the day of the competition. The obstacles include ground obstacles that could include ramps and speed bumps that are <25mm high that the robot should be able to traverse, and red obstruction obstacles that include cylinders and walls which are >400mm high that the robot should manoeuvre around. Each Team has its own home base 300 x 400mm in size within the arena surrounded by a ~10mm high rim to prevent any deposited food packages from rolling out of the base.

The “food packages” all look outwardly identical, a golden steel cylinder of diameter 50mm and height 70mm with an annular groove to facilitate gripping. Each package is either 0.5, 0.75 or 1.0kg in weight with the location of each package changing each round.

This report covers the description of the chosen concept design of the mechanically operated magnet. The report covers the further development of this concept and evaluates it against the chosen requirements from the conceptual design report (CDR). The Report splits the design into 4 parts; Collection Mechanism, Searching Mechanism, Locomotion Mechanism and Software Architecture. Any further developments that are planned are also described against the project requirements.

3 Design Description

3.1 Method of Operation

The robot initially implements the basic wall avoidance function while it searches for a package using a combination of the ArduCam and the upper and lower IR distance sensors. The robot turns on the spot to identify a package. If no package is found it travels towards the most open area. Once a package is detected the robot begins to hone in on the package until it is within the range of the IR photodiode array. The array allows the robot to fine tune it's positioning on its final approach to the package. Once the package is close enough to the array that the magnet can take hold the magnet is activated. The collection mechanism and package is then rotated towards the storage area and the magnet released, dropping the weight into the storage area. The collection mechanism is then returned to its original position and the robot continues its search for more weights.

The navigation mechanism is still under development so the exact method of detecting a weight and the process that the robot executes is not finalised.

3.2 Collection Mechanism

The collection mechanism contains the parts required to accurately acquire the food packages and load them safely in to the storage bay. It can be broken into two main sub-systems: package acquisition and storage. Neither of these sub-systems will provide feedback, they will be actuated by the Arduino when a package has been detected right in front of the robot.

3.2.1 *Package acquisition*

This sub-system includes the magnet that latches onto the packages (which are made of iron), as well as the servos which rotate the magnet. It is ultimately responsible for picking up a package and transferring it to the storage bay. The package must be directly in front of the magnet for it to be attracted by the magnetic field, but it can be in any orientation.

The main component of this sub-system is a mechanically operated magnet that can be switched on and off by rotating the magnet 90° within its housing. This is actuated by a HXT900 micro servo mounted to the top of the magnet. The housing for the magnet is bolted to two DRS-0101 Herkulex smart servos. These servos hold the magnet level, 25 mm above the ground so that its magnetic face is parallel with the side of a food package. When activated, the smart servos can rotate the magnet and any attached packages through an angle of 110° about a horizontal axis. This allows the sub-system to latch onto a package and rotate it to a position where it can slide off the face of the magnet and into the storage bay. Figure 3.1 below shows a Solid works model of the whole sub-system while Figure 3.2 shows the range of motion of the magnet.

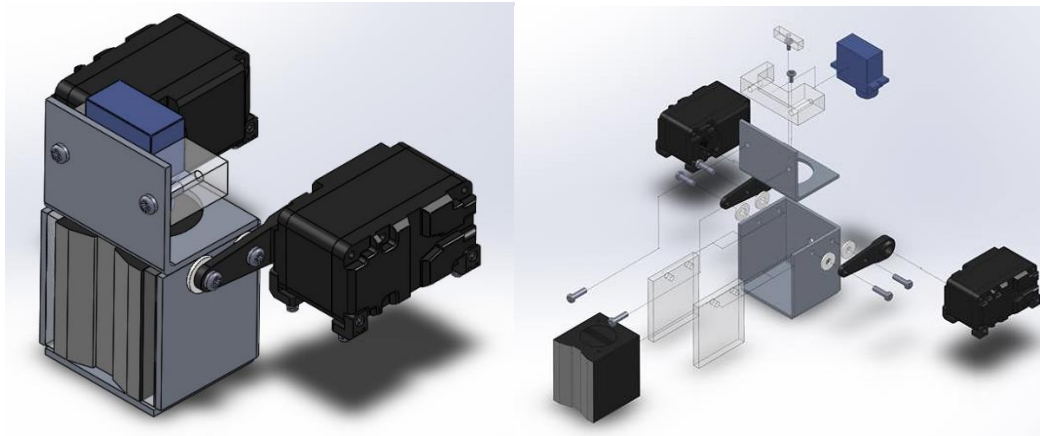


Figure 3.1: CAD model of package acquisition sub-system and its exploded view

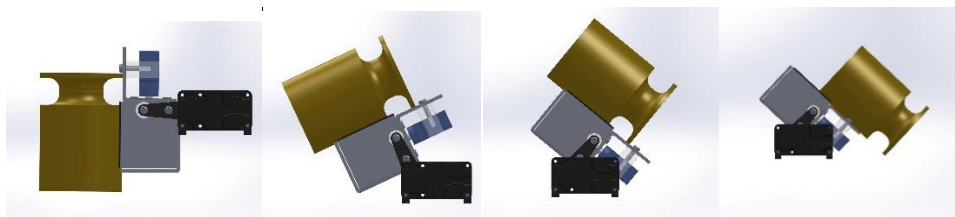


Figure 3.2 Example of package collection

The HXT900 micro servo is shown in blue, while the two DRS-0101 Herkulex smart servos have been shaded black. Figure 3.3 illustrates how the magnet operates in software as described above.

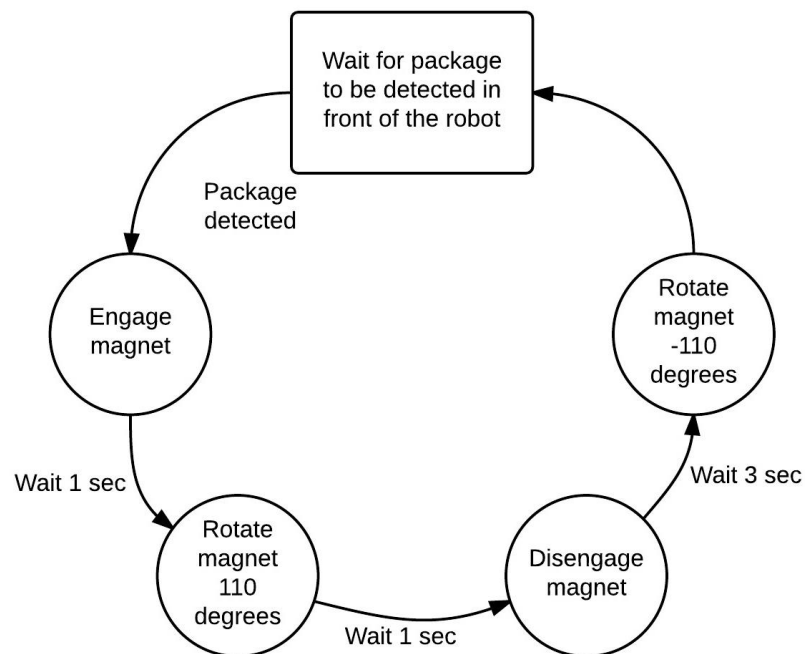


Figure 3.3: Flow Diagram of Magnet Operation

As shown in Figure 3.3, there is no closed loop control, or any method of confirming that a package has been collected, but this is an area for further development.

The micro and smart servos all connect directly to the Arduino via standard RJ11 cables, they do not provide any feedback so they are operated entirely with open loop control. The Arduino actuates them by writing the desired angle to the servo, which is relative to its current angle. The desired angles written to each servo are pre-defined as constants in the software.

3.2.2 Package storage

The storage sub-system is composed of the sloped storage bay area. Its sole function is to contain packages and prevent them from accumulating at the front of the storage bay and potentially blocking the magnet.

The storage area itself is simply the empty cavity inside the chassis, below the Arduino. The primary and only component is the specially sloped plate that forms the bottom of the storage bay. It is a single sheet of aluminium specially bent and mounted to cause the packages to roll towards the back of the robot and away from the magnet. This will in theory prevent packages from accumulating at the magnet and preventing new packages from being deposited in the storage bay. In addition, a triangular length of metal has been fitted to the plate in the spot where packages land once they slide off the magnet. This piece of metal is designed to topple packages to the left or right of the storage bay rather than land upright just behind the magnet. Figure 3.4 shows a conceptual design for the plate. Refer to Appendix E for a detailed schematic of the plate

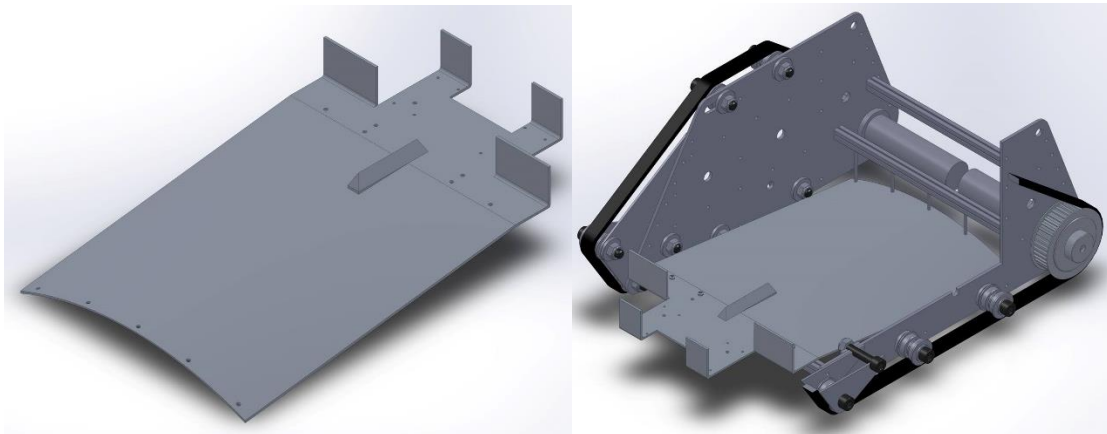


Figure 3.4 Storage plate and storage plate in cutaway of chassis assembly

This design for the storage bay was chosen over its alternatives because it is a passive, purely mechanical solution. It does not require actuation, sensors or any kind of input or feedback which should make it a very robust and reliable solution. In addition to holding packages, the storage tray also functions as a mounting point for sensors and the package acquisition sub-system, i.e. the magnet and smart servos.

3.3 Navigation Mechanism

The navigation mechanism contains the sensors required to avoid obstacles and accurately detect the presence of food packages in the arena. This includes detecting food packages far away from the robot as well as ‘homing in’ on packages that are nearby. There are a variety of sensors which can be divided into 3 main sub-systems: the ArduCam, infrared distance sensors and infrared photodiode array

3.3.1 ArduCam

The ArduCam is used as one of the secondary mechanisms for detecting obstacles. Due to the limited processing power of the Arduino Mega ADK, the ArduCam is only used to detect the location of the red coloured obstacles in the robot's field of view. At this stage it is not used to detect food packages, though this may be extended in the future.

Image Acquisition

Images are retrieved from a hardware buffer on the ArduCam board over the SPI bus. This is a trivial task as there are third party libraries available for the Arduino which automate the process of configuring and capturing images from the camera. Images can be retrieved in either BMP or JPEG format.

At this stage, images are retrieved in bitmap format. This takes a very long time as the available libraries provide no control over the size of the image returned – however they are already in the correct format which means there is no post-processing/decompression required before the data can be used. This process takes approximately 450ms to take a photo and transfer it to the Arduino.

One option currently being explored to decrease this time is configuring the camera to return images in JPEG format. The time to take a photo and transfer it to the Arduino is approximately 160ms in this mode. This mode is considerably faster, but the data returned is not in a format where it can be readily used. The conversion process is typically considered out of reach for an Arduino as decoding a JPEG is computationally intensive. Further investigation is required to determine if this is a viable option.

Image Processing

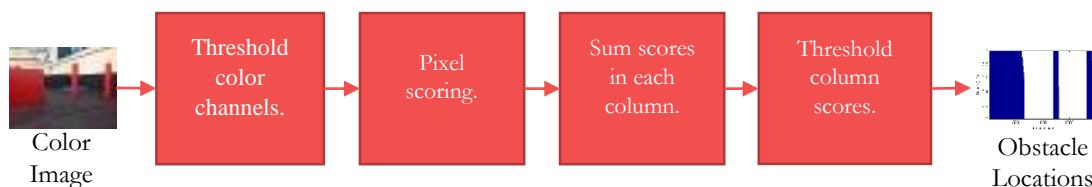


Figure 3.5: Image processing steps to detect red obstacles

Each of the blocks in Figure 3.5 represents a distinct step in the image processing process used for images captured by the ArduCam. The algorithm has been designed with implementation on a microcontroller at the very forefront and requires minimal memory usage as it makes use of the hardware buffer on the ArduCam board to remove the need to store any of the image in memory. Each pixel is simply read in and then discarded, storing only the sum of the scores for each column in memory. The column score is stored as a `long` data type, and so the approximate memory requirement for the image processing software is equal to 32 bytes times the horizontal resolution of the image. This means that images up to approximately 240 pixels wide can be safely processed without using all of the Arduino's limited memory.

The image processing steps are as follows:

1. Threshold colour channels.

The Arduino will read in the R, G and B values for a single pixel. A threshold is then applied and values below a lower bound are set to zero, and those above an upper bound are set to 255. If the value is between these bounds, it is interpolated in to the new range.

2. Pixel scoring.

After any thresholds have been applied, a “score” is calculated for each pixel. This is representative of how red a pixel is (though the algorithm can be tuned to detect other colours). The score is calculated with the following equation:

$$S = q_1 R^2 + q_2 G^2 + q_3 B^2$$

The coefficients q_n have been determined experimentally and are tuned so that the red objects in the arena receive a high score (above 0), while any other object will receive a negative score and not be visible. This is evident in Figure 3.7.



Figure 3.6: Unprocessed test image.

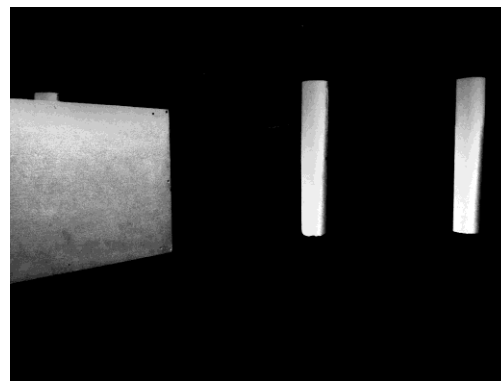
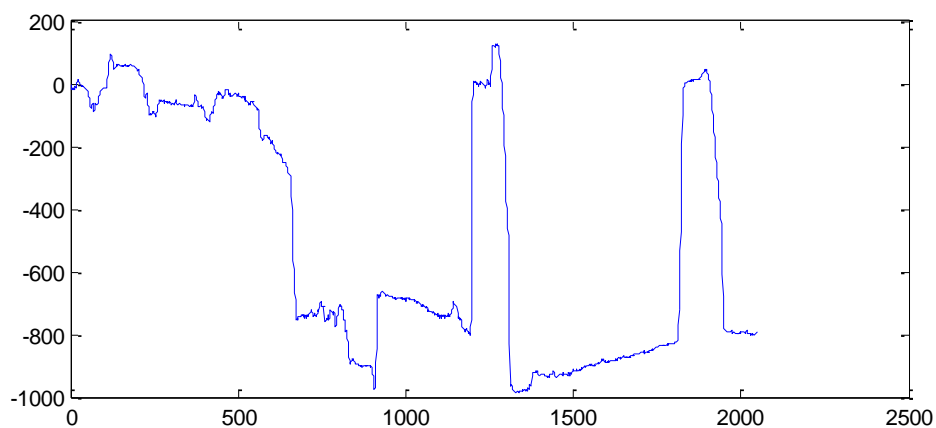


Figure 3.7: An image generated by applying the score function to the original test image.

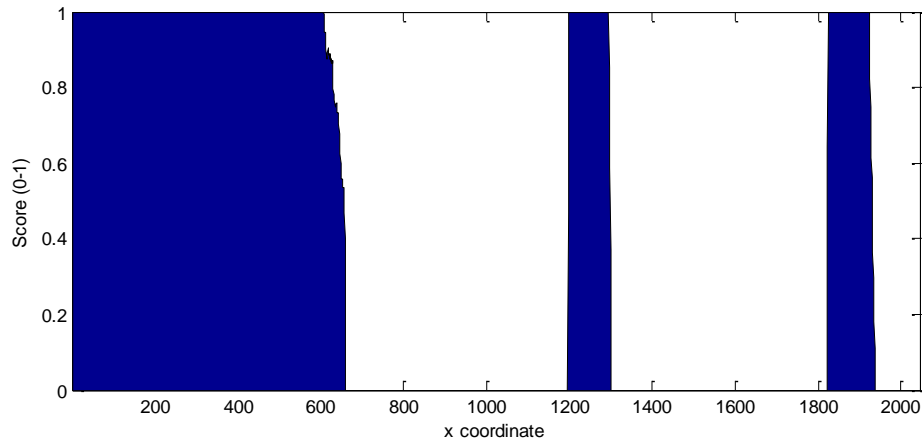
3. Sum scores in each column of the image.

The scores for all the pixels in a given column are then summed and recorded in to a 1-D vector representing the likelihood that there is a tall red object in each column.



4. Threshold column scores.

The sharp peaks in the figure above represent the presence of a red obstacle in that column. By applying a threshold with suitable upper and lower bounds, the graph can be scaled to a 0 or 1 to indicate the presence of an obstacle in a given column.



When this process is complete, it provides a reliable measure of where the red obstacles are relative to the front of the robot. Responses to additional test images have been supplied in Appendix G. The information obtained from this process can be used to support the robot's decision making process when determining what direction to turn to next.

A similar filtering method in order to identify food packages is under investigation. However the colour profile of a food package is not as uniform which may present challenges in effectively filtering the image.

3.3.2 Infrared distance sensors

The infrared distance sensors are used to detect walls and tall obstacles. They have a narrow field of vision and this makes them especially suited to detecting

The IR distance sensors are used to calculate the distance between the sensor and an object. Because they have a narrow field of vision this is the only application they will be used for. This sub-system is still being developed so only a brief overview will be covered.

Three IR sensors will be used in total, two medium range sensors and one long range sensor, the medium range sensors will be mounted along the front of the robot on servos which will allow them to rotate. The long range sensor will be fixed in place on the front of the robot. Figure 3.8 shows the planned cone of vision for each sensor, note that the range of each sensor is much larger than depicted.

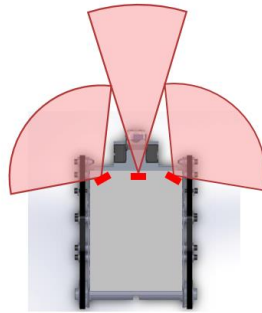
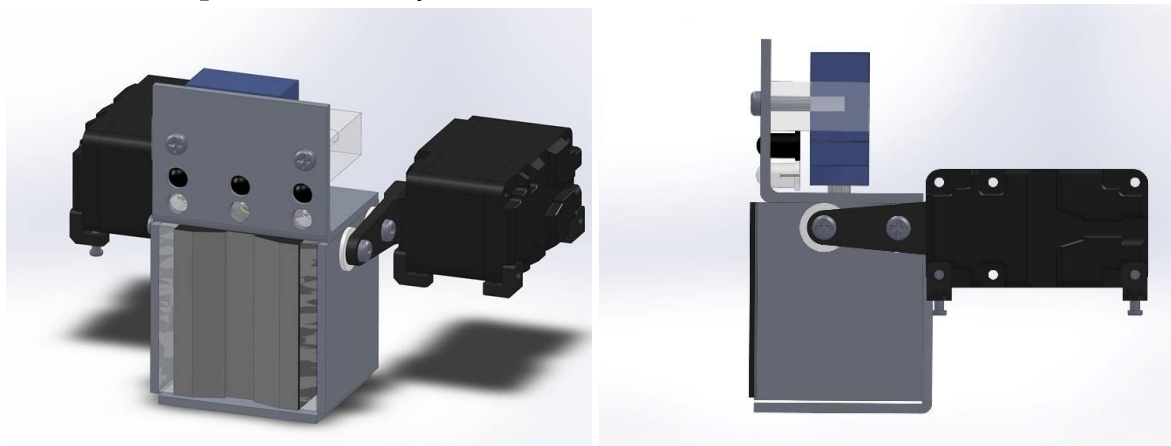


Figure 3.8: IR distance sensor cone of vision

As illustrated, the sensors on the side cover the flanks of the robot while the long range sensor covers anything directly in front of the robot. The distance to the nearest obstacle, and the angle of the sensor will be used in conjunction with the ArduCam to identify an objects distance to the robot and the angle between the object and the robot. These three sensors will all be mounted on the top of the robot, approximately 180 mm above the ground.

3.3.3 Infrared photodiode array



The infrared photodiode array is a tool used to assist the robot in correctly aligning the magnet with food packages on approach. The array consists of three IR LED and photodiode pairs. The photodiodes are connected to an ADC on the Arduino and the measured value is inversely proportional to the amount of reflected IR. This value can be used to infer the presence of a reflective object such as a food package in front of the magnet.

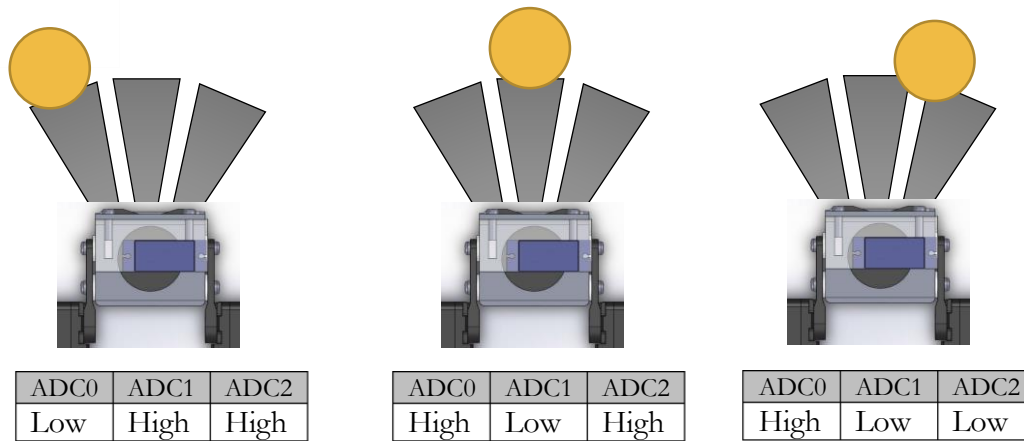


Figure 3.9: Three possible package placement scenarios in front of the IR array.

When the food package is correctly positioned in front of the magnet, the ADC on board the Arduino reads a low value for the central photodiode and higher values for the outside ones. This information can be used to determine the location of the food packages relative to the magnet and make course corrections up until the last moment. A closed-loop PID controller is currently under consideration in order to implement this control scheme. By using the deviation of the food package from centre as the error, a PID controller could easily control the steering signal of the robot's tracks in order to allow the robot to effectively home in on food packages it is approaching.

A circuit diagram for the photodiode array is available in Appendix I.

3.4 Locomotion Mechanism

The locomotion mechanism consists entirely of the chassis and two 28PA51G Low noise DC motors, which is used to move the robot and navigate obstacles.

To increase the amount area available to store the packages, the chassis was rearranged to an up-side down configuration with its long side on the bottom. This setup also has the added benefit of decreasing slip between the belt and the drive wheel due to the drive wheel now being located on the bottom of the robot, adding the weight of the robot to the friction force between the drive wheel and belt.

However, the chassis configuration introduced two problems; the robot could no longer climb over ground obstacles that were up to 25mm high due the track sloping diagonally backwards from the front of the robot, and the cross members of the chassis would 'beach' the robot if driven over the ground obstacles due to them being located very close to the bottom edge of the chassis.

To solve these issues extensions were designed to add to the front of the robot that sloped the tracks diagonally forwards of the robot, rather than backwards allowing the robot to drive over obstacles 25mm and greater. For dimensioned drawings of the extensions refer to Appendix F. The location of the 2 bottom cross beams was changed to solve the beaching issue. The front beam now clears 25mm above the ground and has the storage plate and mount for the collection mechanism fastened to the top of it. The rear bottom cross beam was raised to 45mm above the ground, however to allow for the sloping of the storage tray towards the back the tray was

fastened to the bottom of the cross beam at height of 25mm from the ground. Figure 3.10 shows the redesigned chassis.



Figure 3.10: Revised chassis design

3.5 Software Architecture

The software for the robot uses an open source real-time operating system called ChibiOS. It supports a wide range of hardware platforms and is well supported on the Arduino Mega ADK. It provides a range of services which have made the software design for the robot much more simple and modular. This project makes heavy use of the built-in scheduler ChibiOS provides in order to ensure tasks ran at the desired rate. This has been shown to work well in practice and all versions of the robot software have used this operating system without encountering any difficulties. In addition to the scheduler, the robot software makes use of the Mutex feature provided by the operating system in order to manage access to shared resources such as the I²C interface.

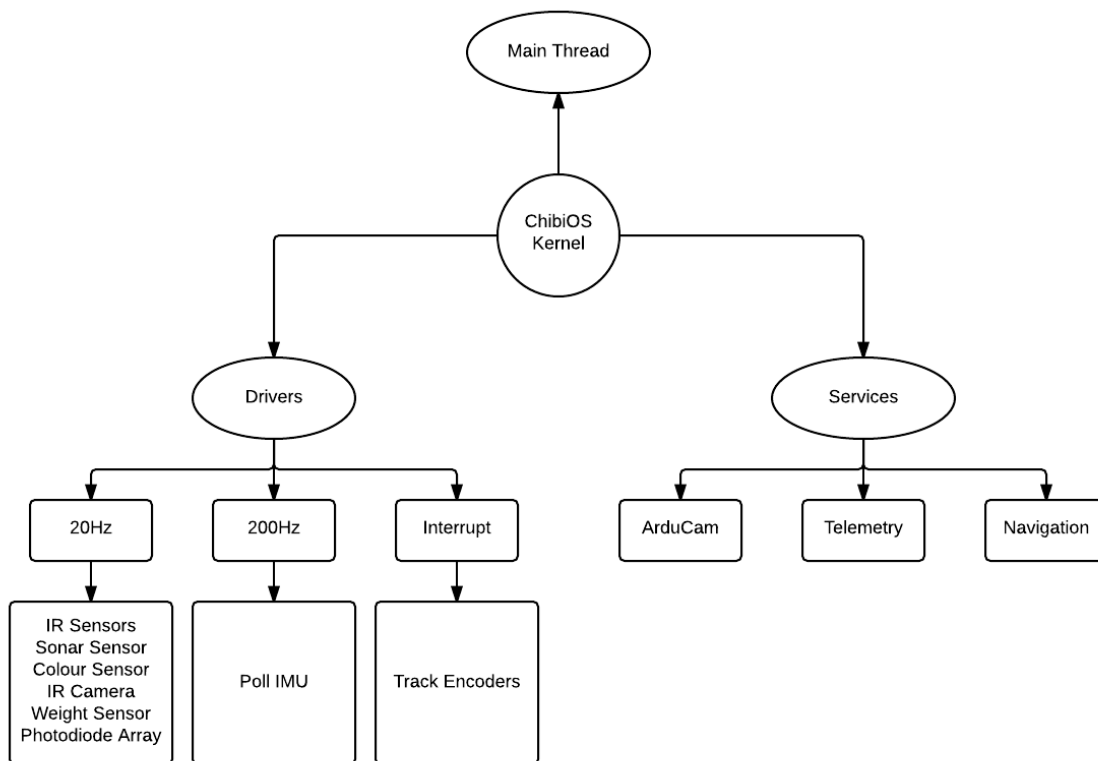


Figure 3.11: Robot Software Architecture.

The program is split in to three main types of subsystems.

3.5.1 The Main Thread

The Main Thread is responsible for all high-level decision making the robot needs to make. It never interfaces with the hardware directly and instead controls the robot by making calls to the different drivers in order to read from sensors or control actuators. By processing this information, the main thread can determine the appropriate course of action and instruct the drivers to carry this out with minimal code.

3.5.2 Drivers

Drivers form the Hardware Abstraction Layer (HAL) of the robot. They provide the interface between the hardware and the main thread. They expose a simple interface and allow for the caller to read or set a value without needing to understand the underlying sensor dynamics or

perform its filtering. This is advantageous because it keeps all code for a specific sensor or actuator in one place, increasing maintainability and simplicity of the overall program.

Drivers will often be run by a number of “management” threads. These are shown on Figure 3.11 and are used to ensure that the drivers update at a certain rate independent of the main thread. This means that the colour sensor, for example is automatically read and the value stored 20 times a second. This means that should the main thread want the present value of the sensor, it does not need to wait for a read to take place (this process can take some time) and can simply read out a value from memory. This improves the overall performance of the program.

Some drivers do not need to run inside management threads such as those for the Smart Servo as they work by simply issuing packets over serial. They have not been included on the diagram above.

3.5.3 Services

Services provide data processing capability to the robot. They are used to process information from the sensors and generate positional information for example. The navigation service is used to integrate the information received from the inertial measurement unit, track encoders in order to dead-reckon the position of the robot, as well as calculate it using the Wii-mote camera to calculate the position of the robot relative to the central IR tower.

Another use of services is for sending telemetry. If all drivers and services were allowed to send debug information over the serial link back to a computer at any time, it could quickly cause the program to slow or experience issues if a context-switch occurred in the middle of sending a serial packet. Instead, all messages are added to a FIFO queue and the telemetry service will empty the queue as CPU time is available.

4 Evaluation

With the major elements of the robot in the final stages of development, the functionality of the robot can be assessed to ensure that it is performing to the specifications laid out in the CDR. In this section, each testable parameter will be discussed and evaluated on a case-by-case basis. Not all of the requirements stated in the CDR are testable, as the robot is not yet fully functional. Therefore only the requirements that have been tested will be discussed.

4.1 Results of the Functionality Assessment

The tests in the functionality assessment and the results are shown below in Table 4.1

Table 4.1: Functionality assessment results

Description	Pass/Fail
Robot can move.	Pass
Robot can move forwards with variable speed.	Pass
Robot can manoeuvre.	Pass
Robot can manoeuvre autonomously.	Pass
2 or more different sensors connected to controller with some detectable output that is a function of the sensor output.	Pass
Control of non-drive motors.	Pass
Robot can collect a package.	Pass

Robot can manoeuvre autonomously with some intelligence	Pass
Robot can specifically detect/identify defined targets such as coloured HQ or packages and reject others.	Pass
Robot can recognise/identify a package in the arena that was not placed directly in front of it, while under autonomous control.	Pass

To pass all of the tests set in the functionality assessment the robot was connected to an Xbox 360 wireless controller to allow the user to drive the robot around the arena and activate each autonomous mode. The controller's analogue thumb stick allowed for variable speed in both directions as well as variable speed during turning. Each trigger of the controller functioned as the switch for the two autonomous modes. The first mode was a wall following mode that utilised the three upper IR distance sensors configured as shown in Figure 3.8. The Robot would drive forward until it got to a certain distance from a wall or obstacle. A series of if statements within a while loop were used to keep the robot at a certain distance away from the wall. If the robot was too far away from the wall it would turn back towards it, allowing it to follow the walls of the arena. The second autonomous mode made the robot turn on the spot until it saw a package then it would drive towards it. The difference between the outputs of the upper middle IR distance sensor and the lower IR distance sensor was used to detect a package. The lower sensor was mounted at the packages height, while the upper far above package height. If the difference was large, i.e. the lower sensor outputs a close reading and the upper a far reading, a package must be obstructing the lower, so drive forward and collect it. The collection mechanisms two Herkulex smart servos and HXT900 micro servo where also mapped to the buttons on the controller. This allowed the robot collect and store packages on board. A colour sensor was added to detect the different colours of each HQ. The RGB values of each HQ colour was recorded and when the colour sensor read values within tolerance of these values the smart servos LED would change to match the current HQ's colour.

4.2 Collection Mechanism

In the CDR for this project, the functional requirements state that the robot must be able to collect 7 packages in 5 minutes. An implicit part of this requirement is that the robot must be able to:

- Collect a package with a mass up to and including 1 kg.
- Collect and store a package within a minimum amount of time.

In addition, the robot must obviously have the storage space for 7 packages.

To ensure that the servos could lift a 1 kg mass, load calculations were completed and compared to the stall torque of the servo. The magnet assembly has a mass of 206 g, and with a 1 kg mass will exert a force of 0.072 Kgm on the servo. A single smart servo has stall torque of 0.12 Kgm, which yields a Factor of Safety (FOS) of 1.67. This FOS was deemed to be inadequate, so it was determined that two smart servos should be used to rotate the magnet, which when combined yielded a FOS of 3.34. The full load calculations are given in Appendix B. When implemented in reality, the smart servos can easily lift a 1 kg package, and in testing thus far have never failed to do so.

To ensure that the robot is capable of collecting 7 packages in 5 minutes, there must be a minimum time required to collect a package. In a worst case scenario, the robot cannot, or will not, move while the package is being collected. As this sub-system is concerned only with collecting packages directly in front of it, the time between the magnet engaging with the package and dropping it into the storage bay must be small. An experiment was conducted where a package was placed directly in front of the robot, the magnet engaged, rotated disengaged to drop the package into the storage bay and finally rotated back to its original position. This process was repeated 30 times and recorded to give an average time taken to successfully collect a package and return to the original position. The average time taken was 3.52 seconds. Because the entire experiment was controlled by the Arduino, the average was almost identical every time, fluctuating only by 0.1-0.2 seconds either way. Any deviation from the average is most likely to due to human error in timing. Refer to Appendix C for raw data.

As specified in the CDR, the robot must be capable of storing 7 packages on-board. This means that the surface area of the storage bay must be equal or greater than the surface area covered by 7 packages. The full storage area of the bay is 55065mm^2 . In a worst case scenario, all collected packages are lying on their side and so have a surface area of 3500mm^2 each, giving a total maximum possible surface area of 24500mm^2 . This is less than the total area of the storage bay, and arbitrary testing by loading packages at random into the storage bay confirms that there is enough space.

4.3 Navigation Mechanism

At present the navigation mechanism has the least functionality of all the various mechanisms. The primary components are all still under development so they do not currently meet any of the project specifications relevant to navigation or autonomous control.

4.4 Locomotion Mechanism

In the CDR for this project, the functional requirements give several requirements for the manoeuvrability of the robot, namely the following:

- The arena walls or the floor outside the arena may not support any of the robots weight.
- The robot shall function on a single charge for greater than 10 minutes.
- The robot shall be able to negotiate all obstacles within the arena while driving forwards or backwards.
- The robot must be able to drive over and turn on top of all obstacles that are up to 25mm high
- The robot must be able to navigate obstacles with a payload of up to 7 kg
- The robot shall be capable of picking up food packages which are placed against arena walls or obstacles. This includes food packages in the corner.

The chassis and motor design complies with the first two requirements as it is not supported by the arena walls or the floor outside the arena, and is able to operate with no losses to performance for greater than 10 minutes on a single charge (enough to last two rounds). To test the manoeuvring capabilities of the chassis, the robot was made to navigate the arena in a series of trials designed to test the effects of an increasingly heavier payload on the robot's mobility. In each trial, the robot had to travel a certain distance with a set payload, and the time taken to

move the required distance was recorded. Each trial was repeated 10 times to average the results. The three trials were:

1. Travel in a straight line for 2 m
2. Travel over a 25 mm high obstacle for 1m
3. Rotate 360° on the spot

The raw time data from each trial is available in Appendix D

It was found that the payload had little effect on the speed of the robot over flat ground. As seen in Figure 4.2 the robot had an average time of 7.45 seconds over the entire test with little deviation even beyond the maximum payload of 7 kg.

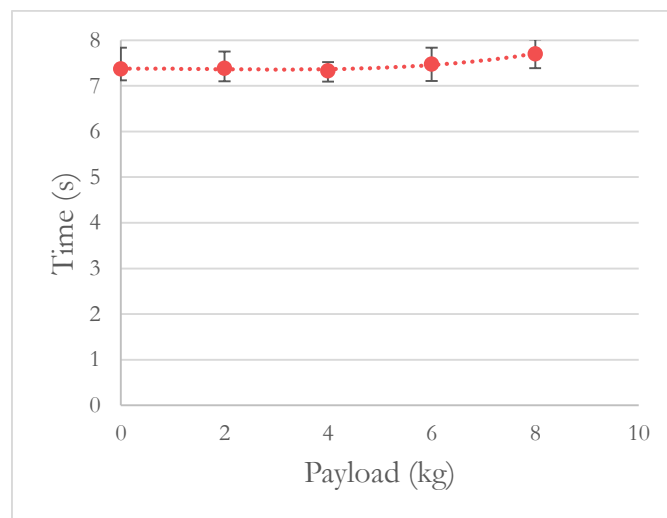


Table 4.2: Time to traverse 2 m

A similar result was seen when rotating on the spot, as illustrated in Figure 4.3

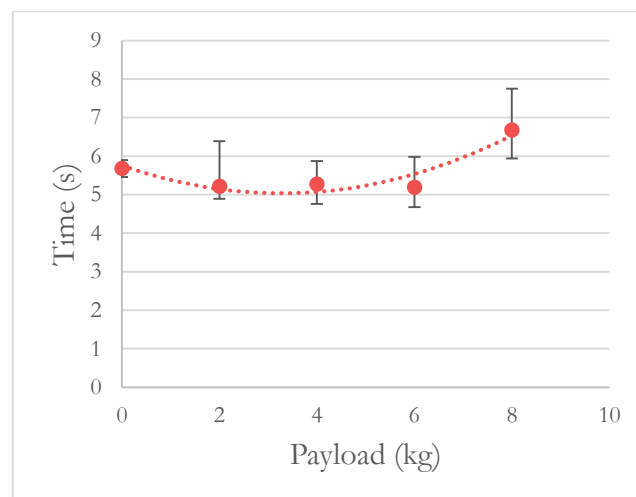


Table 4.3: Time to rotate 360 degrees

From this data it can be concluded that the payload has negligible effect on the average land speed of the robot. However, when crossing obstacles the payload made a significant difference, as shown in Figure 4.1

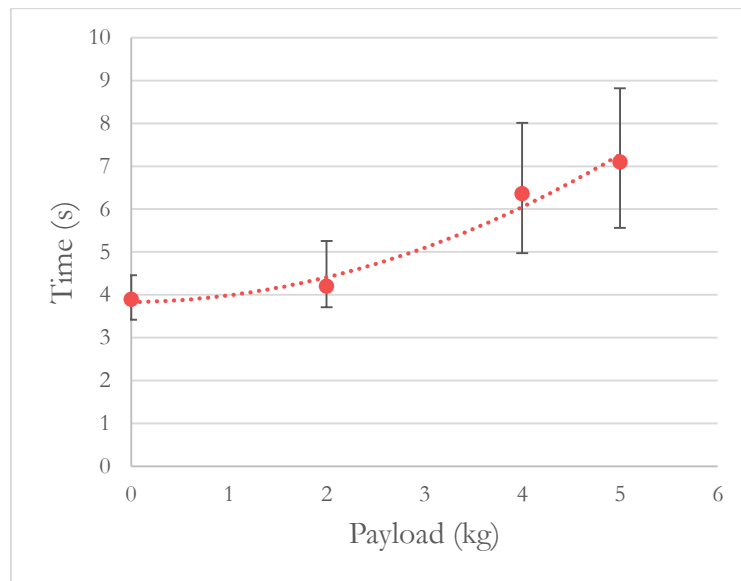


Figure 4.1: Time to traverse a 1 m obstacle

Notably, the robot occasionally failed to climb over an obstacle at all when loaded with a payload of 5 kg or more. And with a payload of 6 kg or more the robot completely failed to traverse any obstacles. This is below the required payload specified in the CDR, so this is a mandatory area for improvement. However the robot does meet the other functional requirements as it is able to successfully access and collect payloads that are located in the corner as shown in figure 4.2.

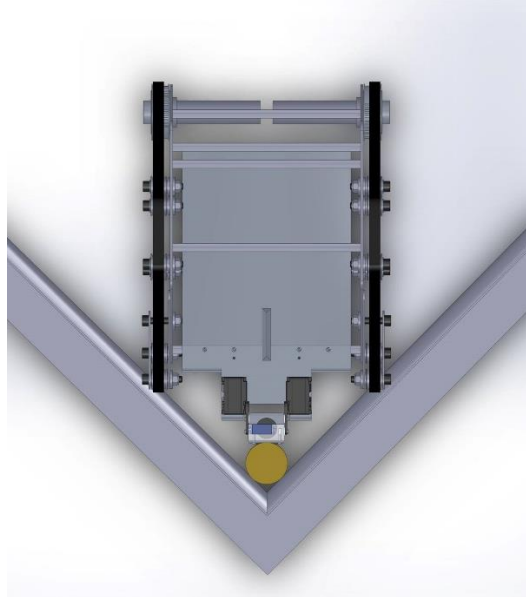


Figure 4.2 Collection of package located in corner

4.5 Software

In the CDR for this project, the following requirements that relate to software were identified.

- The robot firmware shall implement a *scheduler* so that it is trivial to have the robot perform multiple tasks simultaneously and perform all operations in a timely manner.

The use of a pre-existing real-time operating system has greatly simplified the implementation of a scheduler and the robot is able to perform all the tasks required of it in a timely manner. In the latest build of the code, the CPU load factor has not exceeded 0.54 in testing. This was calculated by recording the idle counter when the robot is doing nothing and comparing it to the value of the idle counter when all drivers and services are running. The idle counter is simply a counter which is incremented every time the idle thread runs. If this counter reaches zero, it is a sign that the CPU does not have enough time to service all requests and too great of a load is being placed on it. A CPU load factor of 0.54 implies the CPU is able to process all requests in a timely manner, thus meeting the first requirement.

- All sensors shall be implemented in such a way that any module reading its value need not understand the underlying sensor dynamics.

The sensor interfacing has been done through the use of a hardware abstraction layer. This means that all calls to sensors in the program do not need to understand how the sensors work and can simply make a call, while the underlying driver for that sensor handles all the heavy lifting. This requirement has been met.

- In case of a fault, the robot shall recover and resume normal operation within 10 seconds.
- The robot shall be capable of automatically resetting itself in case of a fault.

This requirement has not yet been addressed, but it is envisaged that the built-in watchdog timer will be used in order to reset the robot in case the software crashes or hangs while operating.

- The robot shall operate the same in both the Mechatronics Design Room C212 and the NZi3 foyer where among other factors, the lighting conditions and the starting orientation (with respect to compass bearing) differ.

This requirement has not yet been addressed specifically, but all software is being written with this requirement in mind. Further testing will be required to determine whether the lighting conditions affect the ArduCam image processing techniques used, the colour sensor or to test for any possible interference on any of the sensors from the different environment.

5 Further Development

There are several aspects of the robot that do not currently meet the performance or functionality requirements laid out in the CDR. In most cases this is because the robot is still in development and does not have full functionality. Any requirements that have not been met for other reasons will be discussed in this section, in addition to any other areas for development that have been added since completion of the CDR.

5.1 Current Requirements

- The robot shall be capable of negotiating all obstacles within the arena with a full 7kg payload.

Currently, the robot is capable of carrying packages and traversing obstacles, but it cannot carry more than 4kg and successfully navigate an obstacle, as discussed in Section 4.3. In order to meet the CDR requirements, the robot must be able to traverse an obstacle with a 7 kg payload. Two solutions have been proposed to solve this issue caused by the beaching of the chassis side plate. The addition of more idlers to the chassis will reduce the amount of the side chassis that can get stuck on the obstacle. A second option of increasing the bottom idler diameter will increase the clearance between the side plate and the obstacle.

- The robot shall maintain knowledge of its position within the arena at all times. This will be done by combining information from all available sensors.

This may be useful to have but not necessary to the success of the robot. If this is able to be implemented in further development of the robot it could be a very beneficial tool used as part of the searching mechanism.

- The robot shall maintain a count of the cumulative weight of food packages aboard the robot. This could be done by measuring the weight of all newly picked up packages, or the weight of the entire container where packages are deposited.

The cumulative weight within the storage tray could be easily measured by the addition of a load cell. The load cell would be used to fix the storage tray to the rear cross member. This addition would also allow the robot to recognise when a package is successfully collected. Limitations of this could be that the load cell is not sensitive enough to measure the increase in weight if the storage tray is too strong, i.e. it will not deflect enough to yield a result when a package is added.

- The robot shall be capable of picking up a food package in the upright orientation on the first attempt in 90% of cases.

This requirement needs to be amended. During testing of the collection mechanism we have realised the robot is able to collect food packages in the upright position on the first attempt in 100% of cases. It has also been noted we have also realised that the robot cannot pick up toppled packages when it approaches them from the side. However, the robot is able to pick up toppled packages on the first attempt in 100% of cases if it approaches from the front or back of the package. It is proposed that the robot should be able to recognise the direction that the package is lying and attempt to approach the package from the easiest angle based on its current knowledge, be it the front or back of the package. The new requirements are as follows:

- The robot shall be capable of collecting a food package in the upright orientation on the first attempt in 100% of cases.
- The robot shall be capable of collecting a toppled food packages from the front or back of the package on the first attempt in 100% of cases.
- The robot shall be able to distinguish the orientation of a toppled food package from an upright food package.

5.2 Additional Developments

Currently, the robot cannot pick up packages in every possible orientation of the package. As shown in Figure 5.1, there are three possible package orientations, an upright orientation and two side orientations. The packages will be placed in the arena in the upright orientation by default, however the robot, or an opposing robot may knock them over, requiring the robot to be able to pick up packages in either of the side orientations. At present the magnet has a 100% success rate in picking up packages in the upright or first sideways orientation (the left and middle images in Figure 5.1). However when collection packages in the second sideways orientation (the far right image in Figure 5.1) the package rolls off the magnet, and fails to be collected. Therefore it is necessary to develop a way for the magnet to be able to collect a package in this orientation. A solution could be the addition of a lip protruding from the bottom of the magnet housing to give enough resistance to the rolling so that the magnet can hold.

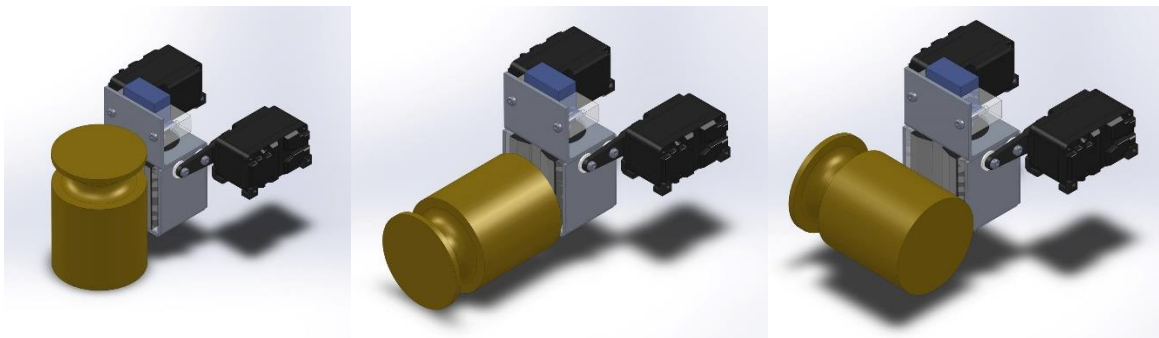


Figure 5.1 Upright orientation, first sideways orientation, second sideways orientation

6 Contribution Statement

6.1 Matthew Young

- Report writing
- CAD modelling of robot, including technical drawings
- Robot performance testing
- Collection mechanism development
- Materials costing

6.2 Hamish Black

- Report writing
- Robot performance testing
- Chassis development

6.3 Chris MacEwan

- Software development
- Report writing
- ArduCam and IR photodiode array development
- Acquisition of magnet and photodiodes

7 Appendices

7.1 Appendix A: Bill of Materials

This list contains all the significant components currently present as part of the overall design. All components are assumed to have been purchased using retail prices rather than wholesale prices. Where no price for aluminium parts has been provided, a price of 5c per gram has been applied to find the total price for the component.

Part description	Supplied	Manufacturer/ supplier	Part #	Qty	Cost	Total cost (\$NZD)
Structural Components						
Chassis tank track	Y	Bestorq	880-8M	2	10.12	20.24
Chassis body	Y	Ullrich	N/A	2	18	36
Chassis cross beam	Y	Ullrich	N/A	5	3.25	16.25
Chassis extension inner plate	N		N/A	2	4.7	9.4
Chassis extension outer plate	N		N/A	2	1.75	3.5
Perspex 300*300*5	Y	Dotmar	N/A	1	6	6
Electronic Boards						
Arduino Mega ADK	Y	Arduino		2	120	240
I/O board	Y	UC		1	40	40
Dual 12A motor driver	Y	DFRobot	DRI0003	1	96	96
Power regulator	Y	UC		1	12.9	12.9
Battery protection	Y	UC		1	14	14
ArduCam	Y	Arduino		1	36	36
Sensors						
IR long range dist sensor	Y	Sharp	GP2Y0A02YK0F	1	17	17
IR medium range dist sensor	Y	Sharp	GP2D12	2	17	34
IR short range distance sensor	Y	Sharp	GP2D120	1	18	18
Servos						
Smart servo	Y	Herkulex	DRS-0101	2	50	100
Micro servo	Y	Herkulex	HXT900	1	4	4
12V low noise DC motor	Y	DFRobot	28PA51G	2	53	106
Standard servo	Y	hexTronik	HX12K	2	12	24
Fasteners						
M3 5 mm screw	Y	Mitre 10	N/A	4	\$3/10	1.2

M3 10 mm screw	Y	Mitre 10	N/A	26	\$3/10	7.8
M3 15 mm screw	Y	Mitre 10	N/A	12	\$3/10	3.6
M3 20 mm screw	Y	Mitre 10	N/A	8	\$3/10	2.4
M8 40 mm hex head bolts	Y	Mitre 10	N/A	14	\$29/40	10.2
M8 1.5 mm thick washer	Y	Mitre 10	N/A	60	0.25	15
M8 2 mm thick washer	Y	Mitre 10	N/A	20	0.28	5.6
M8 hex head nuts	Y	Mitre 10	N/A	20	0.12	2.4
Miscellaneous						
Magnet	N	Unknown	N/A	1	9	9
Smart servo arm	Y	Herkulex	N/A	2	0	0
M8 Bearings	Y	DFRobot	608RS	24	1	24
Drive wheel	Y	Unknown	N/A	2	10	20
Switch	Y	SonarPlus	ST0335	1.8	2	3.6
IR diode	N	RS Components	665-5322	0.3	0.295	0.0885
IR Photodiode	N	RS Components	699-7600	1.6	3	4.8
					TOTAL	942.98

7.2 Appendix B: Load Calculations for Collection Mechanism

These are the full calculations for the FOS of the smart servos. The distance from the package centre of mass (COM) to the axis of rotation is 65 mm. The distance from the magnet COM to the axis of rotation is 22.5 mm.

$$m_{\text{package}} = 1 \text{ kg}$$

$$m_{\text{magnet}} = 0.306 \text{ kg}$$

$$\text{stall torque} = 0.12 \text{ Kgm}$$

$$\text{load} = 1 * (0.065) + 0.306 * (0.0225) = 0.0719 \text{ Kgm}$$

$$FOS_{\text{one servo}} = \frac{0.12}{0.0719} = 1.67$$

$$FOS_{\text{two servo}} = \frac{0.24}{0.0719} = 3.34$$

7.3 Appendix C: Raw Data for Package Collection Timing

The three columns of Table 7.1 give the recorded values for the time taken for the magnet to engage, rotate, disengage and rotate back to its original position.

Table 7.1: Time taken to collect a package in seconds

3.52	3.61	3.53
3.56	3.51	3.57
3.49	3.52	3.5
3.59	3.47	3.49
3.47	3.59	3.58
3.46	3.52	3.54
3.41	3.54	3.54
3.41	3.58	3.67
3.39	3.47	3.56
3.26	3.63	3.52
Average:		3.52

7.4 Appendix D: Raw Data for Robot Mobility Under Load

Table 7.2: Time taken to traverse 1 m

	Payload (kg)				
	0	2	4	6	8
	7.84	7.51	7.38	7.56	7.91
	7.46	7.31	7.27	7.5	7.73
	7.18	7.33	7.22	7.7	8.02
	7.32	7.5	7.38	7.43	7.71
	7.12	7.32	7.46	7.46	7.52
	7.31	7.75	7.52	7.84	7.56
	7.15	7.56	7.28	7.51	7.88
	7.59	7.21	7.09	7.11	7.42
	7.63	7.3	7.47	7.12	7.39
	7.13	7.1	7.23	7.57	7.86
Average	7.373	7.389	7.33	7.48	7.7

Note that for Table___ 'N/A' indicates that the robot failed to clear the obstacle altogether.

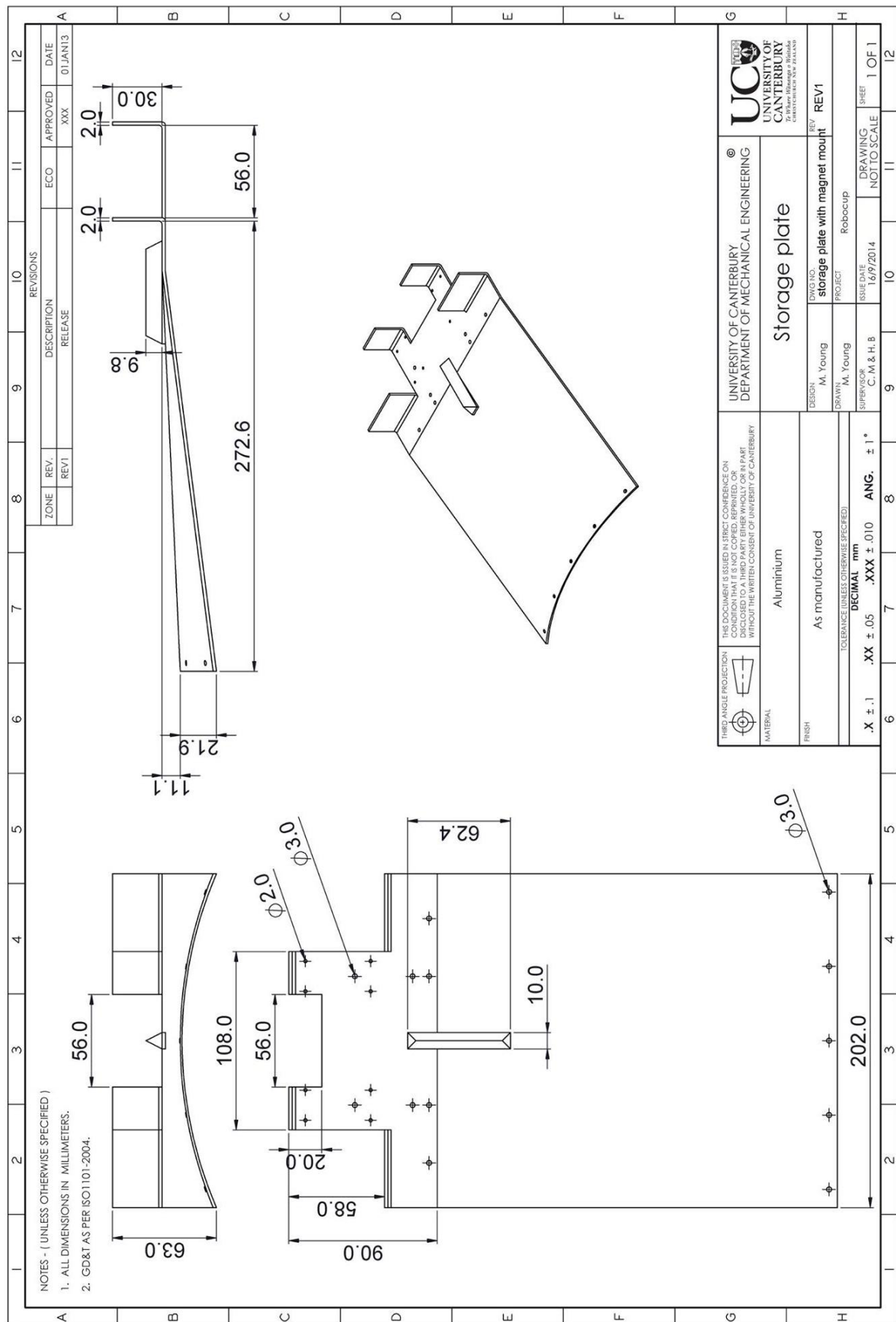
Table 7.3: Time to traverse a 1 m obstacle

	Payload (kg)				
	0	2	4	5	8
	4.13	5.04	5.25	6.92	N/A
	3.64	4.28	6.51	5.56	N/A
	4.46	3.97	4.97	8.82	N/A
	3.85	3.84	6.87	N/A	N/A
	3.42	3.96	8.01	N/A	N/A
	4.09	4.09	7.89	N/A	N/A
	4.01	3.71	5.63	N/A	N/A
	3.51	5.26	6.24	N/A	N/A
	3.58	3.98	5.21	N/A	N/A
	4.26	3.87	7.06	N/A	N/A
Average	3.895	4.2	6.364	7.1	#DIV/0!

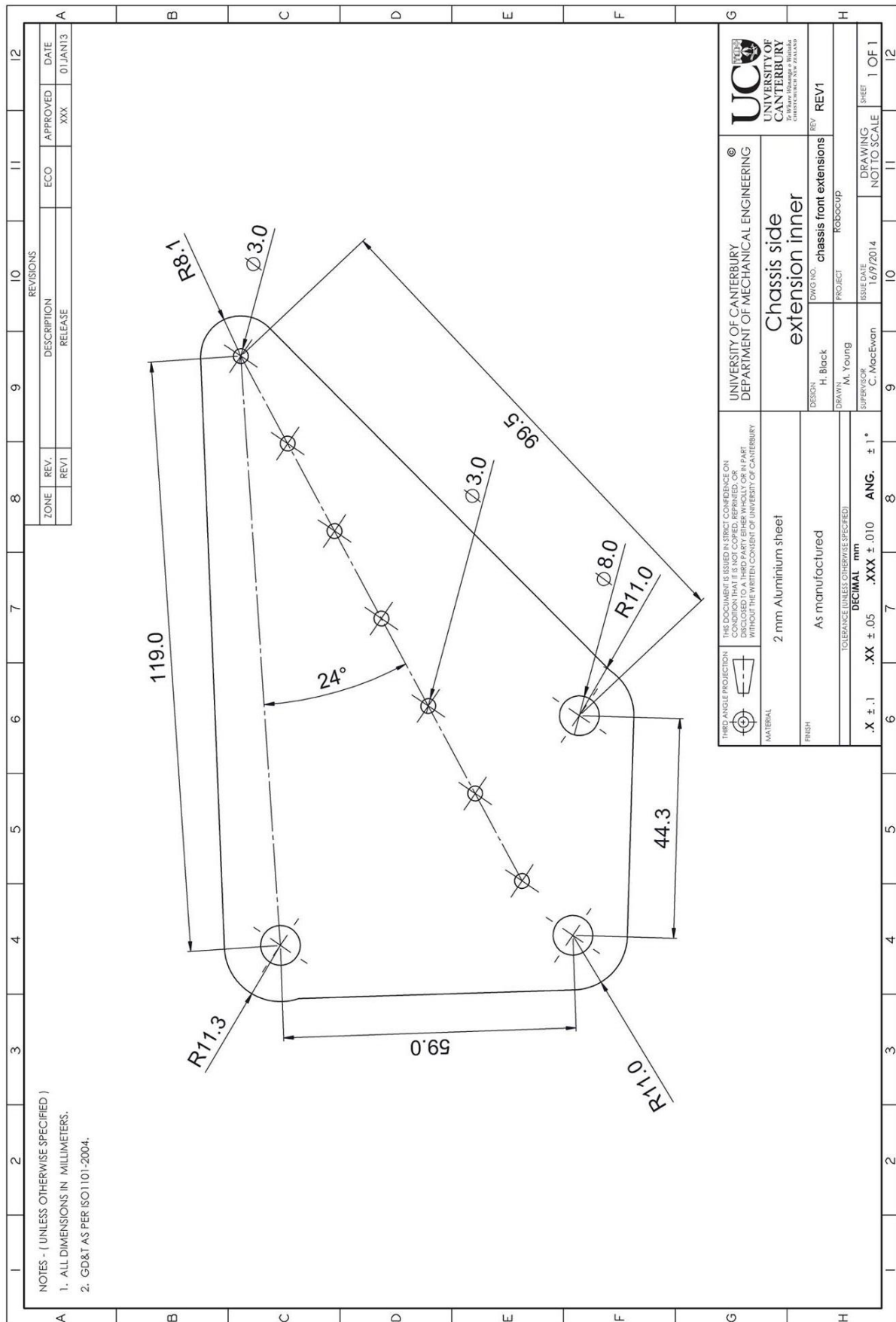
Table 7.4: Time to rotate 360 degrees

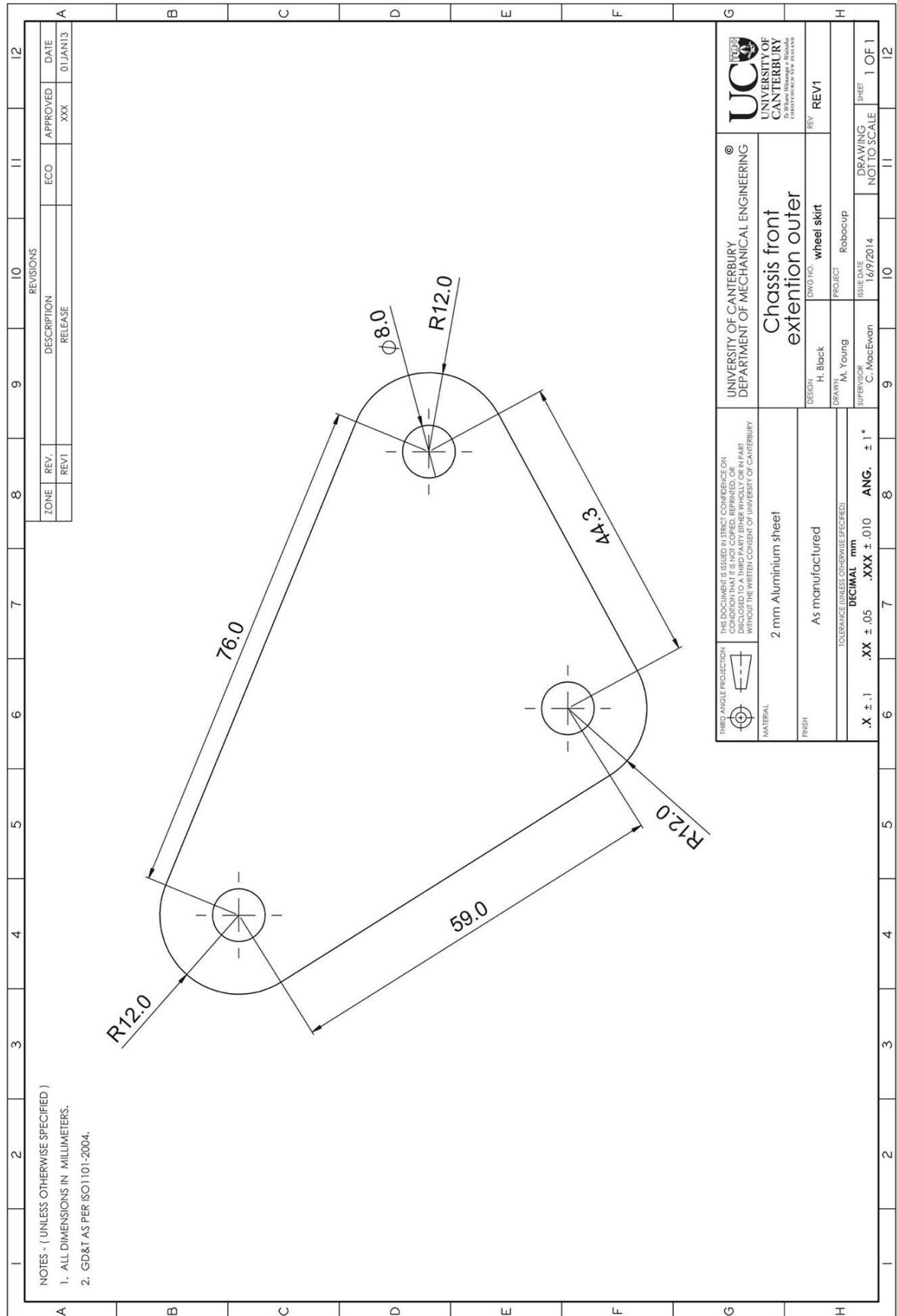
	Payload (kg)				
	0	2	4	6	8
	5.5	6.39	5.18	4.82	5.94
	5.67	5.12	5.43	4.71	6.38
	5.9	4.99	5.39	5.22	6.71
	5.82	4.93	5.62	5.62	6.23
	5.54	5.08	4.76	5.07	7.75
	5.59	5.01	5.29	5.36	7.67
	5.89	5.25	4.98	4.86	6.03
	5.46	4.89	5.87	5.58	6.08
	5.67	5.34	5.14	5.98	7.12
	5.81	5.18	5.06	4.68	6.84
Average	5.685	5.218	5.272	5.19	6.675

7.5 Appendix E: Storage plate schematic

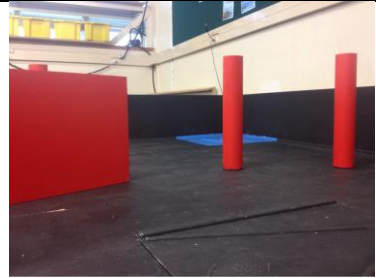
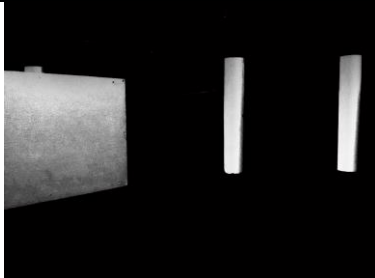
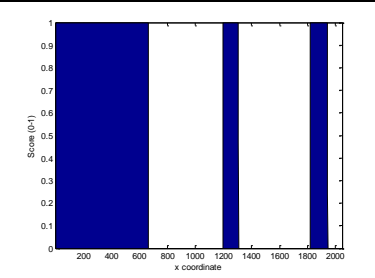
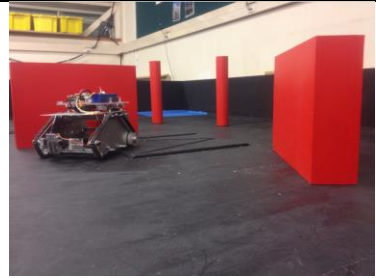

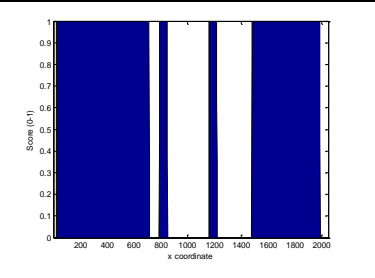
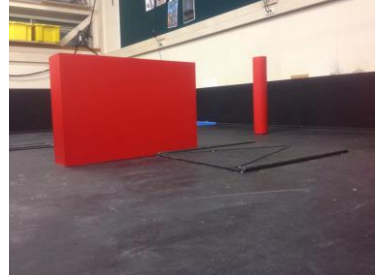
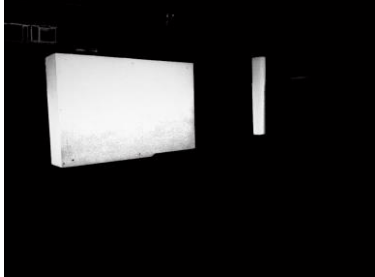
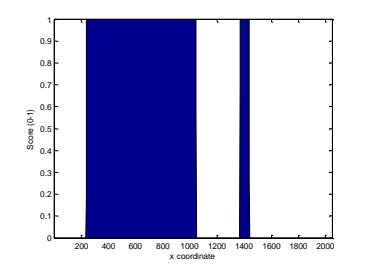

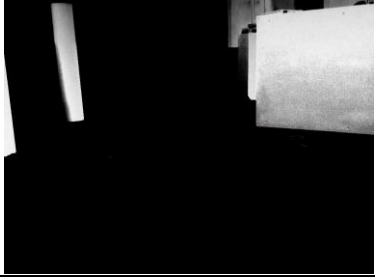
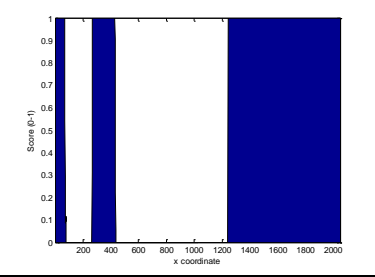

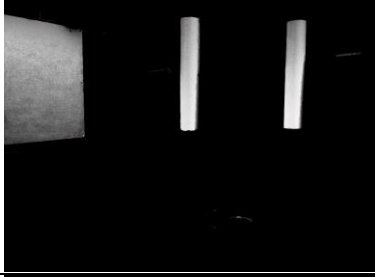
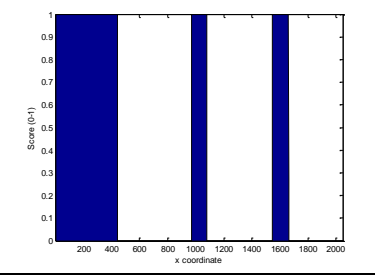
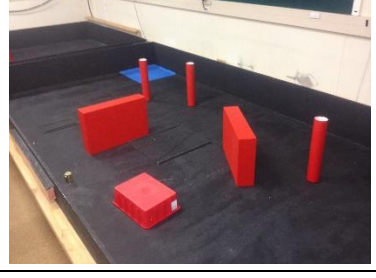
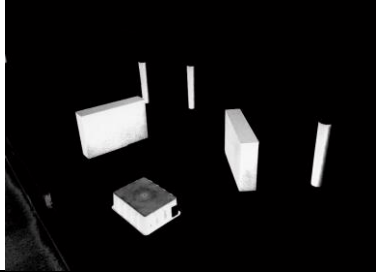
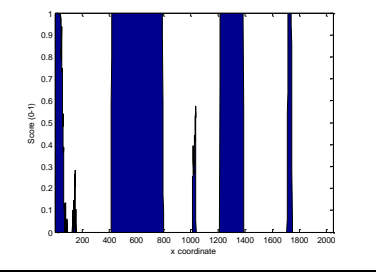


7.6 Appendix F: Chassis extension manufacturing drawings





7.7 Appendix G: Image Processing Test Sets

Input Image	Filtered Image	Objects Detected
		
		
		
		
		
		

7.8 Appendix H: Image Processing MATLAB Script

This image processing algorithm has been tested in MATLAB on a set of test images. The script used for producing these images is shown below.

```
%Image Processing Demo for Robocup
%Author: Chris MacEwan
%Date: 17/09/2014

clear all
close all

%Import the test image and display it on the screen using imshow.
I = imread('test6.jpg');
imshow(I);

%Stage 1: Thresholding.
%Split out the R, G and B components in to separate matrices and apply
%the thresholds.
I_red = mat2gray(I(:,:,1), [120 180]);
I_green = mat2gray(I(:,:,2), [0 230]);
I_blue = mat2gray(I(:,:,3), [0 230]);

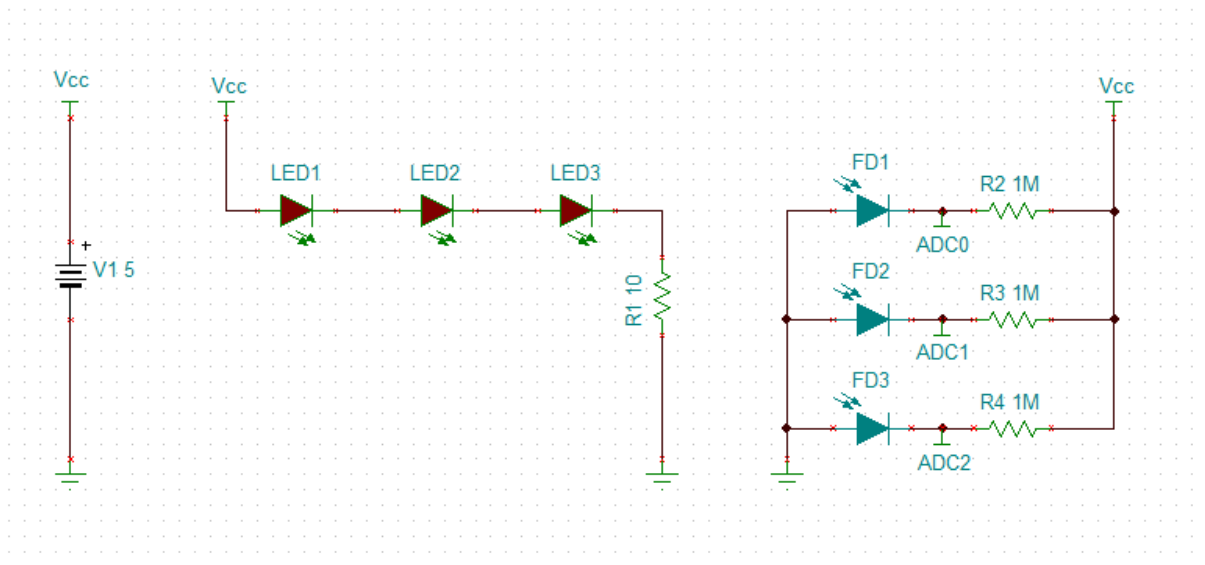
%Stage 2: Scoring.
%Calculate the score for each pixel and display it as an image.
q = [1 -1.3 -1.3];
SCORE = (q(1) * I_red.^2) + (q(2) * I_green.^2) + (q(3) * I_blue.^2);
figure (2); imshow(SCORE);

%Stage 3: Sum column scores.
%Add up all the scores in each column and display a score for each
%on a line graph.
colscore = sum(SCORE);
figure(3); plot (colscore);

%Stage 4: Threshold and perform object detection.
bound_l = -550;
bound_u = -500;
diff = abs(bound_u - bound_l);

colscore = (min(max(colscore, bound_l), bound_u) -bound_l) / (diff);
figure (4); area (1:length(colscore),colscore);
ylabel('Score (0-1)');
xlabel('x coordinate');
```

7.9 Appendix I: Photodiode Array Driver Circuit



This circuit uses the following components:

- 3x LD271 950nm Infrared LED
- 3x BVP10NF 940nm Photodiode
- 3x 1MΩ Resistor
- 1x 10Ω Resistor

The circuit has not been built yet, therefore it's components have not been included in the Bill of Materials.

7.10 Appendix J: Magnet Housing Schematic

