ATLAS

Urban Search and Rescue Robot

Technical Report 2016/2017

Guy Baker | Dan Carmichael | Andrew Gilley | Adam Hong | Oliver Mackinnon | Thandiwe Ngoma | Michael Rajaretnam

Warwick Mobile Robotics

Abstract

The aim of the project was to design and manufacture a functional urban search and rescue robot to revive the competitiveness of the Warwick Mobile Robotics (WMR) team. This robot was named ATLAS.

The method by which this was achieved was through a clear and powerful strategy of ‘form follows function’ and intelligent use of resources such as time, money and people. This strategy was supported by the inception of the ‘Warwick Mobile Robotics 10 Stage Plan’ that guided the team through the key project milestones.

For each sub-system that was implemented into ATLAS, a basic design was quickly manufactured so that basic functionality could be pursued. These basic systems included track and motor control with regard to electronic systems and chassis, tensioning and flipper arms for mechanical systems.

Iterations thereafter came as a result of feedback from critical analysis of current designs and previous designs, as well as literature review. The innovative systems were improvements upon those aforementioned and were structured to act as a guide for future WMR teams to work from. These innovations included the incorporation of sensors for the electronic systems and optimised topology, dynamic suspension, dynamic tensioning and extendable flipper arms for mechanical systems.

Within the allotted project time of 30 weeks, all basic designs were manufactured and tested in terms of manoeuvrability and exploration where applicable. All tests were conducted in compliance with the International RoboCup Rescue League Rules.

ATLAS successfully completed all fundamental movements including forwards, backwards, turning and raising itself up on its flippers. More advanced tests featured obstacles such as 35^{o} and 45^{o} staircases. ATLAS was successful in navigating the former, however, on the latter, mechanical failure occurred as a result of old components which will need replacing.

In conclusion ATLAS was able to perform with higher capability than the previous three years of robots at Warwick. All stages of the ‘10 Stage Plan’ were fulfilled, culminating in a functioning robot and a comprehensive handover document with an abundance of recommendations for future WMR teams to build upon.

Declaration:

We, the authors, hereby declare that the work presented in this technical report relating to “ATLAS – Urban Search and Rescue” is entirely the work of Warwick Mobile Robotics team 2016/17. This is in partial fulfilment of the ES410 Group Project assessment criterion.

[illegible]	
Guy Baker - 1323686	Date
[illegible]	
Dan Carmichael - 1318395	Date
Andrew Gilley	
Andrew Gilley -1024766	Date
Adam	
Adam Hong -1314229	Date
[illegible]	
Oliver Mackinnon - 1211170	Date
[illegible]	
Thandiwe Ngoma - 1310747	Date
[illegible]	
Michael Rajaretnam - 1218887	Date

Acknowledgements

The Warwick Mobile Robotics Team would like to take this moment to thank all of those who contributed towards the inception of ATLAS, our latest Urban Search and Rescue Robot. Without those of you we have acknowledged below, we would have never had achieved such success with ATLAS.

Dr. Emma Rushforth	For her guidance and weekly availability throughout the project.
Jacob Gates	For his amazing commitment as the team's honorary 8th man.
Margaret Low	For allowing us to inspire the next generation of budding engineers.
Edgar Zauls **Stefan Winkzist**	For giving us their time to inspect our designs and share their wisdom.

And of course our respected sponsors for their continued support:

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programing Interface
CAD	Computer Aided Design
COG	Centre of Gravity
CO_2	Carbon Dioxide
ESCON	Enterprise Systems Connection
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
IMU	Inertial Measurement Unit
LED	Light Emitting Diode
LiDAR	Light Detection and Ranging
LiPO	Lithium Polymer
NiCAD	Nickel-Cadmium
Nimh	Nickel-Metal Hydride
PS3	Play station 3
POV	Point-of-View
PCB	Printed Circuit Board
PDB	Power Distribution Board
PWM	Pulse Width Modulation
ROS	Robotic Operating System
SDK	Software Development Kit
SPI	Serial Peripheral Interface
SLAM	Simultaneous Localisation and Mapping
UAV	Unmanned Aerial Vehicles

UGV	Unmanned Ground Vehicle
UUV	Unmanned Underwater Vehicles
USB	Universal Serial Bus
USV	Unmanned Surface Vehicles
USAR	Urban Search and Rescue
VDC	Virtual Design and Construction
VSS	Vertical Spring Suspension
VVSS	Vertical Volute Spring Suspension
WMR	Warwick Mobile Robotics
WMG	Warwick Manufacturing Group

1. Introduction

Disasters - whether they are natural or man-made - occur on a frequent basis all over the world. From earthquakes to warzones, inhabited areas can fall into anarchy as infrastructure crumbles around the population. Both large and small populations can fall victim as they become trapped under debris or exposed to the elements. Where there are victims that need to be rescued, emergency personnel are often dispatched to the scene to aid those in need. They, in turn, risk their own lives and are limited by the human ability to see and act. The drive to mitigate this risk and more effectively locate and assist victims has afforded the development and deployment of robotic systems in the field of Urban Search and Rescue (USAR) Robots [1].

Warwick Mobile Robotics (WMR) - a team of Master's level engineers at the University of Warwick – have been making advances in robotic systems in the field of USAR since 2008. Motivated by the desire to help those in need, the USAR robots produced by WMR have been entered in numerous international competitions, which evaluate their effectiveness in real-world scenarios; to remarkable success. The aptly named USAR robot 'Champion' was the result of three years of innovation at WMR and won three titles at the RoboCup competition between 2010 and 2013 [2]. However, within the last three years, the effort to miniaturise the USAR robot has overcomplicated the development process and resulted in no functional robot. This, coupled with 'Champion' falling out of commission due to prolonged usage, has caused a loss of faith in WMR and a subsequent loss of sponsorship has ensued, making funding a major constraint.

This technical report introduces the latest addition to Warwick Mobile Robotics' USAR Robot fleet – ATLAS. It presents the key development stages that were required to revive WMR's reputation for the year 2016/17. Entailing the management strategy, technical aspects, and proposals for future systems. Finally, the Real-world testing that was used as a measurement of ATLAS' success is documented. The accompanying cost-benefit analysis addresses the financial and social benefits associated with ATLAS [3]

1.1. Team Strategy

This was a fresh start for WMR and the pressure mounted to make a working USAR robot. There were two obstacles to realising this dream, namely funding and time constraints. It was therefore decided to adopt the mind-set of "function over form" and "simplicity before complexity". The strategy was to design all critical components for the robot to function first, and then have them manufactured as soon as possible to resolve any design issues early on. This allowed for testing to be carried out and to leave a foundation for future WMR teams. Having this strategy in mind from the beginning gave the team direction and a clear goal in which to achieve. Proving evidence of ATLAS' competency through testing would allow future years to liaise with potential sponsors, securing much needed funding and support.

In the interest of saving time, the strategy was to learn from the mistakes and successes of previous years as well as performing a literature review into 'state-of-the-art' USARs for inspiration.

As a means of reducing expenses, the strategy was to hold an inventory of the WMR work area. Previous USAR robots such as 'Champion' and 'Cyclone' were also dismantled and their parts scrutinised for re-usability.

1.2. Aims & Objectives

The aims of the ATLAS project were twofold. Firstly, to design and manufacture a functioning Urban Search and Rescue robot, capable of representing Warwick University at international robotics competitions. Secondly, to create an aesthetic platform and comprehensive handover document to our successors, so that future design innovations and sponsorship scouting can begin immediately in 2017.

For the ATLAS team to realise these aims, the following objectives were proposed:

- Decide on ATLAS' anatomy, based on what is available through the disassembly of previous years' robots and literature review.
- Design and manufacture initial designs with an emphasis of modularity so that future years can change and innovate on the base design.
- Design and rapid prototype future design possibilities, so that next years' team have detailed starting points to work from.
- Validate the designs through real-world testing and evaluation against RoboCup competition requirements.

These objectives were further sub-categorised into more detail, and became known as 'Warwick Mobile Robotics' 10 Stage Plan 2016/2017"

1.3. Warwick Mobile Robotics 10 Stage Plan 2016/2017

Figure 1 shows an infographic depicting the 10 stages that the ATLAS team agreed upon. These fell into four milestone phases, known as Specification, Functionality, Innovation and Testing Phases.

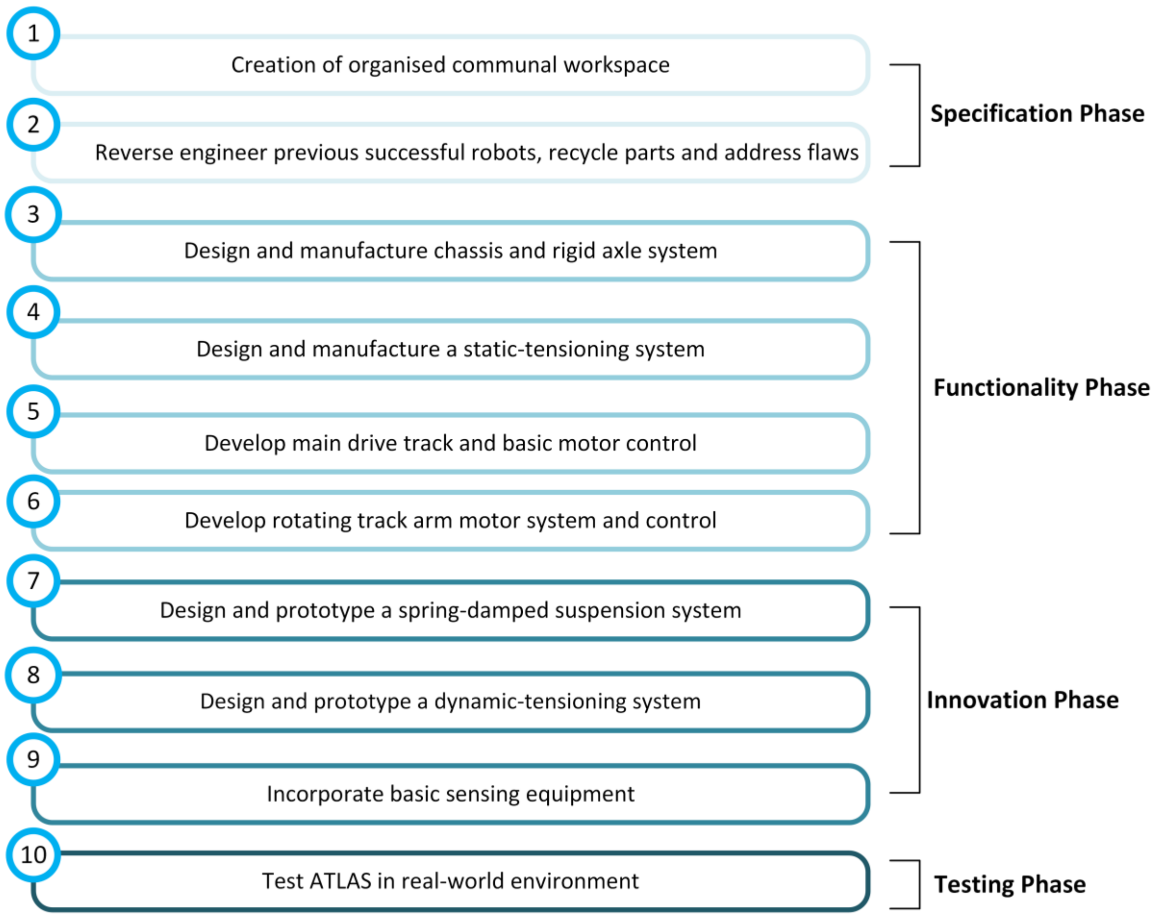

1
Creation of organised communal workspace
2
Reverse engineer previous successful robots, recycle parts and address flaws
Specification Phase
3
Design and manufacture chassis and rigid axle system
4
Design and manufacture a static-tensioning system
5
Develop main drive track and basic motor control
6
Develop rotating track arm motor system and control
Functionality Phase
7
Design and prototype a spring-damped suspension system
8
Design and prototype a dynamic-tensioning system
9
Incorporate basic sensing equipment
Innovation Phase
10
Test ATLAS in real-world environment
Testing Phase

Figure 1: Warwick Mobile Robotics' '10 Stage Plan 2016/17'

Despite the number sequence, the stages of the '10 Stage Plan' could be, and were worked on in tandem. An example of this was the mechanical team undertook stages 3 and 4 in parallel with the electrical team tackling stages 4 and 5 in the 'Functionality' Phase.

The '10 Stage Plan' was the best-case scenario for the project and so long as the team's resources were organised and managed effectively, completing all ten stages was deemed realistic and achievable.

2. Resource Management

With a project of this scale, it was imperative that resources such as time, people and funding were used effectively. This section details the means to which this was achieved through project management.

2.1. Time Management

There were 30 weeks to realise ATLAS and to leave sufficient time for the design, manufacture and testing, strict deadlines for design freezes were implemented. Figure 2 shows the distinctive design phases identified in the "WMR 10 Stage Plan". The date at the bottom of each phase's box indicates the deadline for that phase, and all work in progress should move on to the next phase unless it is being corrected because of feedback.

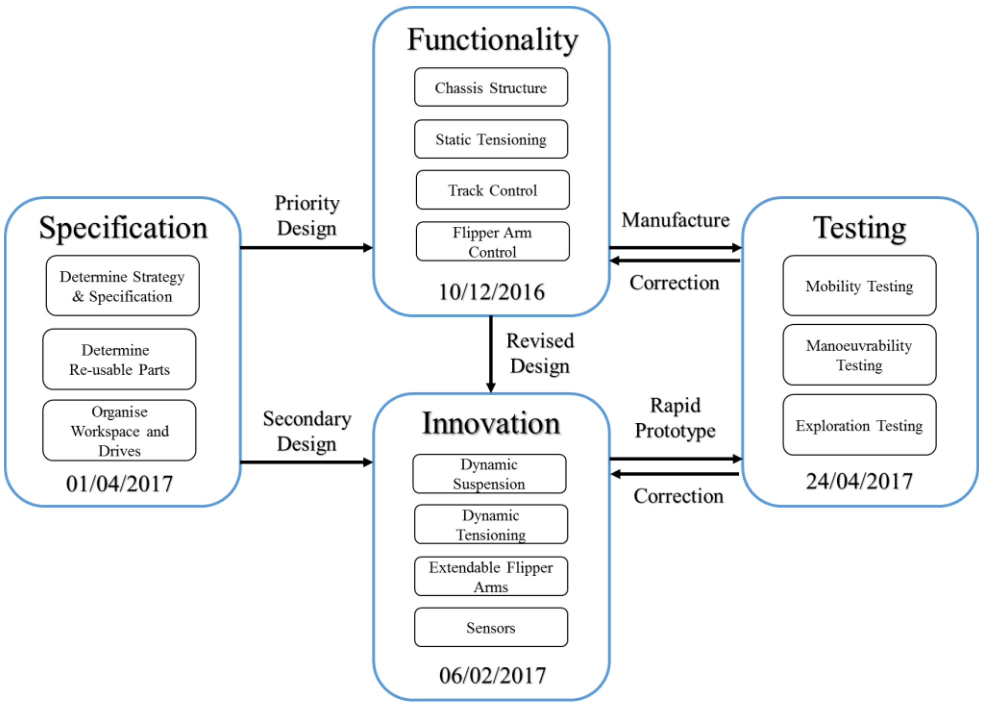


Figure 2: ATLAS Design Phases

Other time management strategies involved the creation of Gantt charts. An example of the Gantt chart used for the organisation of this technical report can be found in Appendix A.

2.2. Team Structure

The WMR is a multidisciplinary team comprised of engineers from the School of Engineering at the University of Warwick. The team were fortunate to include three mechanical engineers, two electrical engineers, one systems engineer and one automotive engineer. There was a variety of skills to be drawn upon and allowed for two sub-teams to be formed; these broadly divided as electrical and mechanical. Each team member was given a specific sub-system of ATLAS of which they were responsible for.

Alongside each team member's specific sub-system, a secondary management role encompassing other important aspects of ATLAS' development was assigned to them.

Project Manager – Responsible for the delegation of work and ensuring project completion.

Chief Design Engineer – Responsible for major design choices and manufacturing.

Media and Marketing – Responsible for maintenance of WMR's website and social media.

Secretary – Responsible for taking minutes at each team meeting.

Health & Safety Officer – Responsible for health and safety documentation and evaluation.

Finance Manager – Responsible for material orders and budget reviews.

Outreach Officer – Responsible for facilitating outreach events such as Imagineering.

2.3. Logistics

Weekly meetings were held with the project supervisor, with all team members expected to attend. Each sub-team leader relayed progress and concerns to the project manager prior to these meetings. This information was collated into a weekly agenda, available for team members to review before discussion in the meeting itself. An example of an agenda can be found in Appendix B.

The meetings were structured in the following way to cover all topics:

1) Individual and sub-team review and status of work completed during the week.
2) Discussion of weekly topics proposed by the project manager and resolution of issues that may have arisen.
3) Delegation of tasks and division into smaller sub-teams for the next week of work.
4) Targets that should be met before the next week's meeting.

It was agreed that all Computer Aided Design (CAD) work be done using SolidWorks and any files relevant to the project should be uploaded to the team's shared Google Drive in the corresponding folders. For communication between team members, WhatsApp was the chosen medium.

3. Motivation

This section describes the need for Urban Search and Rescue robots and why Warwick Mobile Robotics have endeavoured to develop them. It also includes a description of the competitions that drive innovation in the field of search and rescue robotics.

3.1. 'State of the Art' Urban Search and Rescue Robots

Urban Search and Rescue (USAR) robots are designed with the capability to provide support to both responders and victims following natural or manmade disasters and reduce the risk to human life. The idea is that the robots can be used to assess risks and mitigate damages whilst protecting human lives.

Due to the arduous and hazardous nature of the environments in which USAR robots are deployed in, it is important that they be designed to be physically robust with advanced locomotion capabilities. This enables them to scale unstable terrains such as rubble [4]. They are intelligent systems with advanced sensing capabilities that can allow them to map areas. They can provide feedback through merging various sensor readings, which it collates and transmits back to responders monitoring the situation [5]. Search and rescue missions are slow, labour intensive processes under strenuous and dangerous conditions. It has been found that victim mortality increases drastically after the first 48 hours and therefore time a critical factor [4]. USAR robots can help in speeding up this process and so aid in reducing victim mortality rates.

The need for USAR robots has increased exponentially following increased regularity and intensity of natural disasters [6] [7]. USAR tasks are undertaken at great risk to the lives of the first responders. The advantages of USAR robots stems from the fact that not only do they reduce the risk to humans by removing them from the threat zone, they can also traverse environments that are filled with harmful chemicals and take detailed surveillance of the area/environment using an array of sensors.

The first reported instance of robot deployment for aid in a rescue environment was in the attacks on the World Trade Centre in 2001. Figure 3 shows two of the robots brought to aid with the World Trade Centre response units.

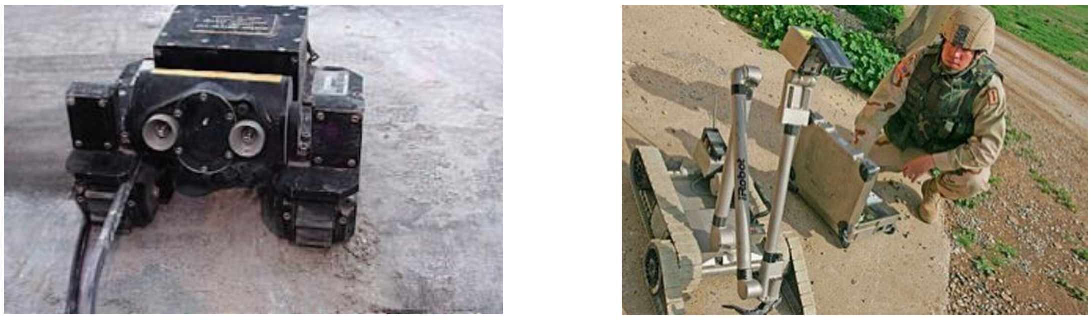

(a) (b)

Figure 3: (a) A micro-Tracs robot, (b) An iRobot Packbot [8]

Since the World Trade Centre in 2001, the frequency that robots have been deployed into the field has increased. When Hurricane Katrina hit the Gulf Coast in 2005, Unmanned Aerial Vehicles (UAVs) were deployed to search buildings and perform structural inspections [8]. Following the earthquake and tsunami in eastern Japan that resulted in the Fukushima Daiichi Nuclear Power Station accident [9], further response robots were requested to aid with surveillance. They were needed to determine radiation levels and inspect reactor damage [10]. These Urban Ground Vehicles (UGVs) were upgraded and altered to meet the requirements of the accident using systems such as articulated arms and Geiger counters [11]. Figure 4 shows a photo of the rescue robots, named Quince, that were used.

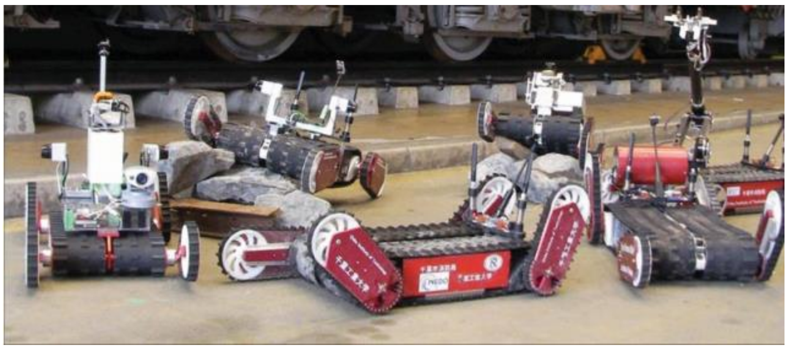


Figure 4: Series of 'Quince' robots supporting different equipment

The unpredictable nature of disaster environments has created a need for a wide range of robotic systems in various forms; each tailored to deal with different scenarios [4]. Each robot must have primary capabilities, normally divided into classifications that include reconnaissance and mapping, search, logistics and in situ medical inspection [12].

The various types of robot can also be grouped in to land vehicles such as Unmanned Ground Vehicles (UGV), aerial vehicles like Unmanned Aerial Vehicles (UAV) and marine vehicles that can come in the form of Unmanned Surface Vehicles (USV) and Unmanned Underwater Vehicles (UUV).

UAVs have been found to be optimal for accidents that cover a large area, being capable of mapping unknown terrain and more easily pinpointing a victim's location [13]. On the other hand, UGVs are generally more useful for manmade disasters that are more geographically concentrated.

Smaller UGVs can fit through gaps or tunnels that are inaccessible to rescuers in search of victims and aid by allowing medical personnel to communicate with them [14]. Larger UGVs can be used to negotiate difficult terrain, create a map of the area using various sensors or help with removing rubble faster than a manned team would be able to [15]. Like previous WMR teams, the WMR 2016/17 Team have focused on creating a tele-operated UGV under the title of 'Urban Search and Rescue Robotics'.

3.2. Warwick Mobile Robotics

Since 2008, WMR have been designing and building USAR robots to compete in the global annual RoboCup competition. Between 2014 to 2016, the team decided to pursue the field miniature USAR (mUSAR). These were named 'Orion' and 'Cyclone' in 2015 and 2016 respectively (shown in Figure 5). These mUSARs are capable of traversing areas previously inaccessible due to natural size restrictions of their significantly larger predecessor, 'Champion'.

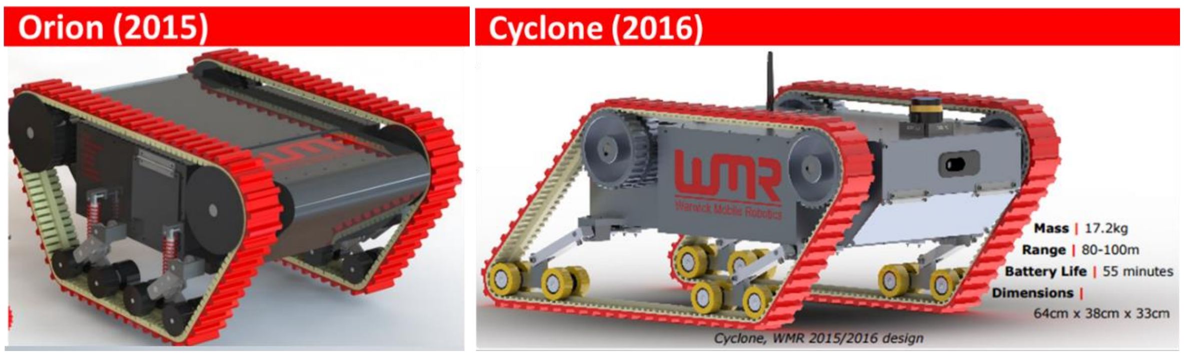


Figure 5: CAD images of previous WMR USAR robots, Orion (2015) and Cyclone (2016)

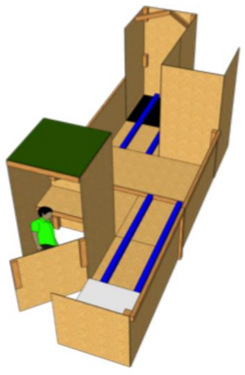
However, Orion and Cyclone did not perform up to expected standards due to limited mobility on rough terrains and incompletion respectively. Ultimately, both failed to enter the competition. After thorough reviews and considerations, the 2017 team has decided to return to a larger USAR robot design based on Champion.

The new robot ATLAS intends to combine the functionality from Cyclone with the versatility of Champion with the goal to compete in RoboCup.

3.3. RoboCup Competition

RoboCup is an annual international robotics competition that aims to promote research and development of robotics through a variety of challenges that mimic real-world scenarios [16]. Among the several Robocop leagues, WMR designs USAR robots to comply with RoboCup's Rescue League. In the Rescue League, robots are deployed for emergency response in a range of simulated urban search and rescue scenarios ranging from simple movements to dextrous manipulation.

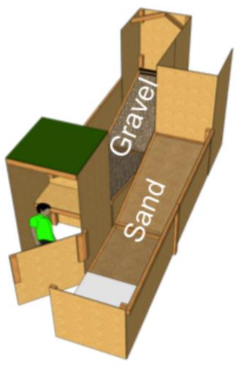
For 2017, RoboCup Rescue have newly implemented a standardised process for measuring the capability of competing robots. According to their new Rule Book [17], the competition is structured into four suites which test a robot's manoeuvrability, mobility, exploration and dexterity. The scope of the ATLAS project was to create an USAR that is capable of competing in the first three suites, as the latter requires the design and manufacture of an articulated arm. There are a total of 20 individual testing bays across the competition, and ATLAS would be capable of competing in the minimum amount of 10. Robots are awarded points for every lap of the bay it can complete within 20 minutes, with penalties induced for any failures or interaction. Examples of the typical bays ATLAS will encounter as shown in Figure 6.



Test: Alignment

Robots must travel in a straight line across 100mm width bars, placed on their outer ground contact dimension.

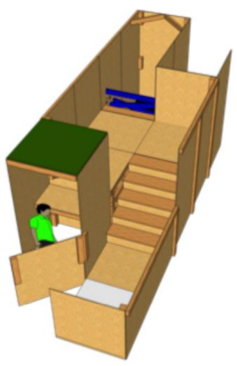
(a)



Test: Sand/Gravel Hill

Alternating 15º hills of sand and gravel will test robots' durability.

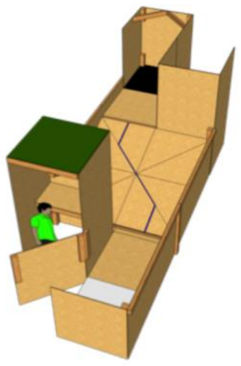
(b)



Test: Stair Debris

35º and 45º stair obstacles with debris in the way.

(c)



Test: Traverse

Robots must climb a 30º incline while closely following the zig-zag pattern.

(d)

Figure 6: RoboCup Rescue Testing Bays (a) Alignment (b) Sand/Gravel Hill (c) Stair Debris (d) Traverse

4. Mechanical Design

The ‘Warwick Mobile Robotics’ 10 Stage Plan allowed for parallel design and innovation to take place between the mechanical and electrical teams. This section details what was learned through literature review and disassembly of previous WMR robots with regards to the mechanical aspects. Each sub-assembly that was chosen to be developed is structured in a way that criticises previous designs, introduces the design that was implemented into ATLAS and finally the innovations that were designed for future years to consider. This was in compliance with Stages 2 to 9 of the 10 Stage Plan for the mechanical elements.

The strategy for mechanical design was devised to ensure a smooth workflow from preliminary design to manufacture. This is shown in the diagram, Figure 7.

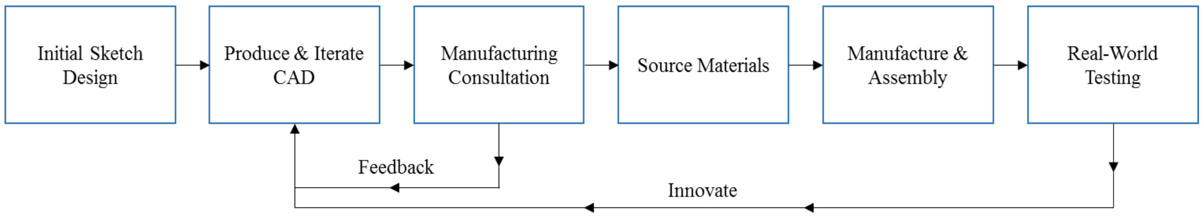


Figure 7: Mechanical Design Strategy Flowchart

4.1. Mechanical Overview

Figure 8 shows labelled diagram of all the sub-systems discussed in this section:

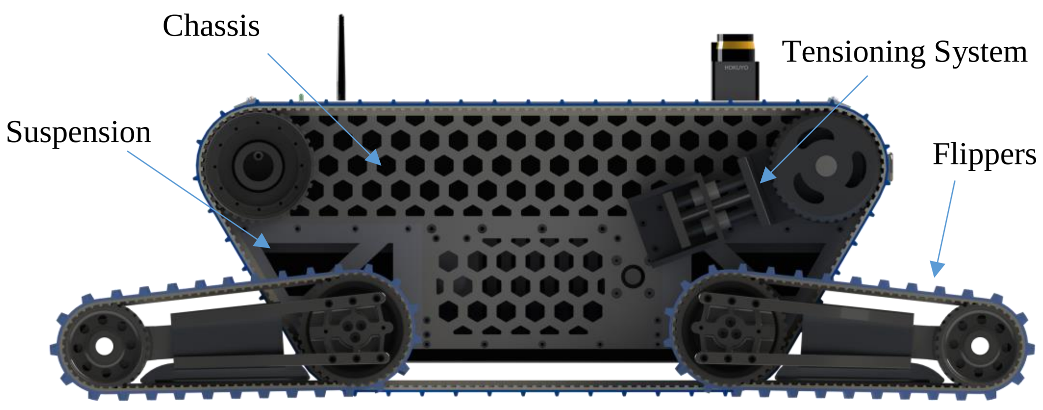


Figure 8: Labelled diagram of ATLAS’s subsystems

4.2. Mechanical Literature Review & Disassembly of USAR ‘Champion’

The 2013 USAR 'Champion' benefited from yearly design iterations since its inception in 2009. The format of having two inner tracks, and moving from a front flipper arm pair to front and rear flipper arm pairs have proven a suitable choice in traversing difficult terrain, winning the

RoboCup ‘best in class for mobility’ award in 2010. The battery location moving from the sides to the front has increased access for their quick and easy removal. Its size and weight offer a sturdy platform upon which to mount an articulated dexterous arm. The flipper arms have performed satisfactorily for a number of years. Whilst the size and weight of the robot do aid with stability and would help in the future with victim retrieval, this has not come without cost. Its retirement came because of repeated structural deformations coming from impacts, leading to bent motor shafts and damaged gearboxes. One of which can be seen in Figure 9.

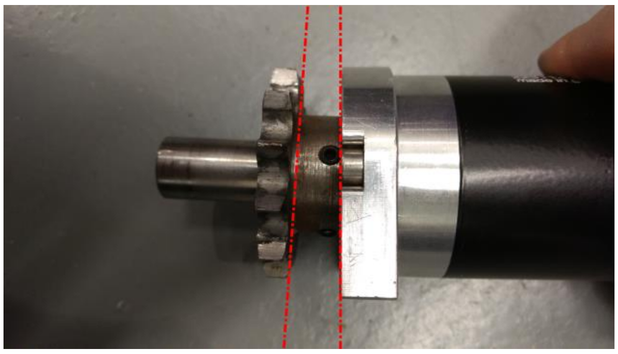


Figure 9*:* Image depicting bent motor shaft of previous robot 'Champion'

The bending was likely caused by sudden impulses experienced when the robot repeatedly fell onto hard surfaces. The flipper arms themselves experience extreme torsion due to their plastic material. The flipper chain fixtures were sub-optimally designed, leading to slack in the chain. This could be a cause for the damaged gearboxes as they experience a surge in torque when the chain becomes taut. The real-world testing of the robot led to an ingress of dirt and debris inside the tracks, unexpectedly increasing track tension and resulted in the chassis sides deforming. With the robot chassis consisting of just several, large pieces, the lack of modularity exacerbated the problem with the deformed chassis.

4.3. Design Choices

As discussed in previous sections, ATLAS is based upon the successful features of previous robots. Its large size increase in comparison to the 2013-2016 robots is key trait that it inherits from ‘Champion’. Although the smaller robots had advantages when it came to moving into tighter areas, the key determining factor behind this choice was the other robots’ difficulty in climbing and navigating through rough and obstructed terrain such as stairs and blocks. Although they were successfully optimised to make the most out of their size and overcome these challenges, in the end they were simply too small to effectively achieve such methods. The second determining factor was the lack of space for practical expansion and improvement, including space for an articulated arm as well as space for control systems and power distribution for newly implemented systems.

The shape of Champion was that of a wedge, which elevated the drive wheels to the back of the robot. Its design was fundamentally stable, but even as effective as it was; it was not without flaws. The fixed positioning of the wheels made both static and dynamic tensioning of the track impossible, this flaw did eventually lead to damage to the system because of trapped debris within the track system. Additionally, due to the lack of tensioning, static or otherwise, the potential to add suspension has been inhibited; therefore, a key aspect of adapting the Champion design was the inclusion of a tension system. Upon deliberation, it was felt by the team that the simplest method to introduce a tension system was the addition of an extra wheel; this approach was also done by WMR teams in 2014/15 and 2015/16. The model for the introduction of the extra tensioning wheel was based upon the concept of modern tanks, where the tension wheel is placed at the same elevation as the drive wheel at the opposite end of the machine. As such, the shape of ATLAS was developed as a trapezium (shown in Figure 10).

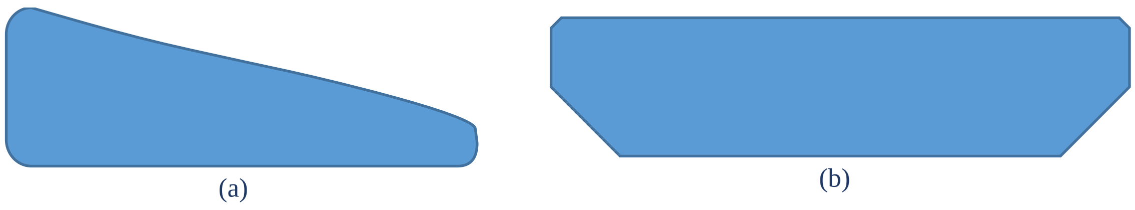


Figure 10: (a) Original Champion’s wedge shape, (b) ATLAS’s new trapezium shape

Another aspect of Champion that was applied to ATLAS was the use of flippers on the lower wheels. Although they were removed from further iterations of the WMR project, it was determined that they were more beneficial to the system and a practical aspect of the robot that should be implemented. They can be used for stabilisation of the robot and an aid in climbing up and down extreme obstacles, as well as increasing traction of the robot when they are moved into a parallel position, increasing the contact level with the ground.

Following the use of a tracked drive system, a dual drive method for the two tracks was selected. Its application is due to its relative simplicity and effectiveness, with one motor driving each track. This allows for increased torque and power over single motor drive systems as well as simplifying the manufacturing and design process.

4.4. Chassis Design & Layout

This section discusses the development of ATLAS' chassis, considering previous chassis designs and makes suggestions for future chassis optimisation.

4.4.1. Critical Review of Previous Designs

The chassis design system that was used as part of Cyclone was a series of mechanical fastenings in conjunction with sheet aluminium plates that could be manufactured separately. The chassis itself successfully underwent intensive light-weighting to optimise the strength and weight, overall a sound design. However, the size and high levels of specialisation rendered improvements and modifications to the chassis difficult despite their claims of modularity.

4.4.2. ATLAS - Chassis Design

As detailed in Section 3.1, the main objective of ATLAS is to enter zones with a substantial threat level to humans, whilst simultaneously carrying a series of complex and delicate components and potential payloads. Therefore, the first and foremost priority is to create a structure that possesses sufficient strength, rigidity and capacity to complete its operation. This is while also paying attention to the following design criterion:

1. The chassis is required to isolate the internal systems from any foreign material that could cause potential damage within the operating region.
2. The robot must be operable in most environment types; urban and rural, inside and outside. The robot's size has thus been inhibited such that it can turn within a space 1.2m [17], whilst being large enough to manoeuvre across large structures and uneven terrain.
3. The chassis must be adaptable and future proof against the introduction of additional components and features. It must therefore have sufficient space or attachment points for easy expansion.
4. A sufficiently designed chassis will improve the process of component positioning and allow for a balanced and low centre of gravity that will assist with ATLAS's operational capability.

In accordance with the proposed design strategy, the priority of design was 'functionality over form', with the aim to improve over time, as such; to expedite the design process, the major functionality of ATLAS is built upon the successful features of previous robots, most noticeably from the robot Champion. This is even down to utilising components from said robot, before improving upon the problems that were found within the original design. Due to the increased size of ATLAS, even in comparison to Champion, who was the largest robot that WMR has ever made, ATLAS's structure was broken down and separated into two smaller sub-structures which in turn are dubbed the Lower and Upper sections.; both of which were designed separately as well as in tandem.

The design of both sections has been achieved utilising simple sheets of material of varying thicknesses, specifically 6mm, 8mm and 10mm; this cuts down on the level of complex manufacturing required and enables easier exchange of components in accordance with the policy of iterative improvements. The whole design is mechanically fastened using button head and socket cap screws, thread sizes of M3, M4 and M5 for the adequate sheet thicknesses.

A consequence of the adding a 7th and 8th tension wheel to system is the original Champion wedge shape design has been modified into a trapezium shape that is reminiscent of a modern tank. Maximising the physical space afforded by the shape without interfering with track mobility, whilst also providing a more balanced weight distribution (Figure 11) .

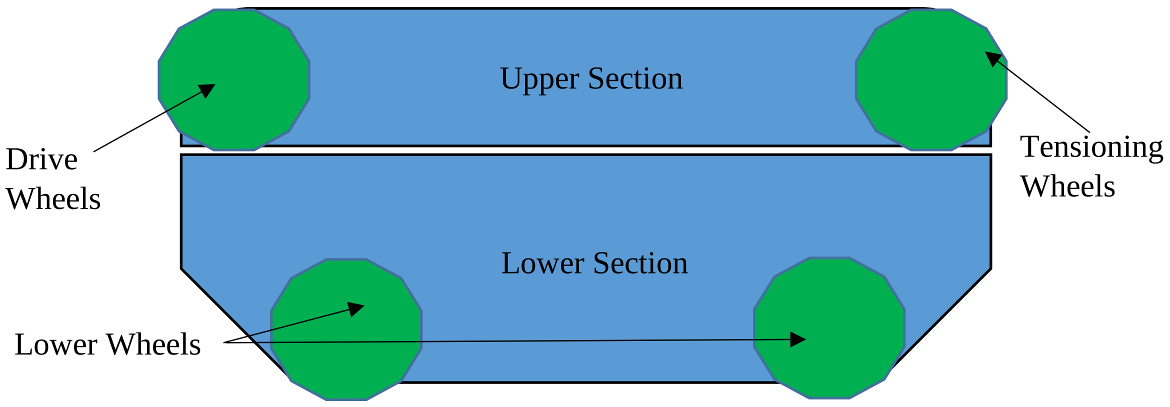


Figure 11: Breakdown of ATLAS's chassis

Lower Section

The lower section is the foundation for the whole robot, encompassing the flipper motors, axles, suspension and supply drop case. It is comprised of a small central box that directly fastens to the central plate and connects the two sections. The suspension system proceeds to connect to the box and central plate (Figure 12).

The lower section is comprised of the larger quantity of heavy material, including the thicker sheets for strength, as well as the elements of the flipper system. The layout is designed with symmetry of the robot in mind, as such, a weight imbalance created by the components is corrected by the equivalent component on the opposite side, resulting in the lower section being much heavier and centrally balanced (Figure 12).

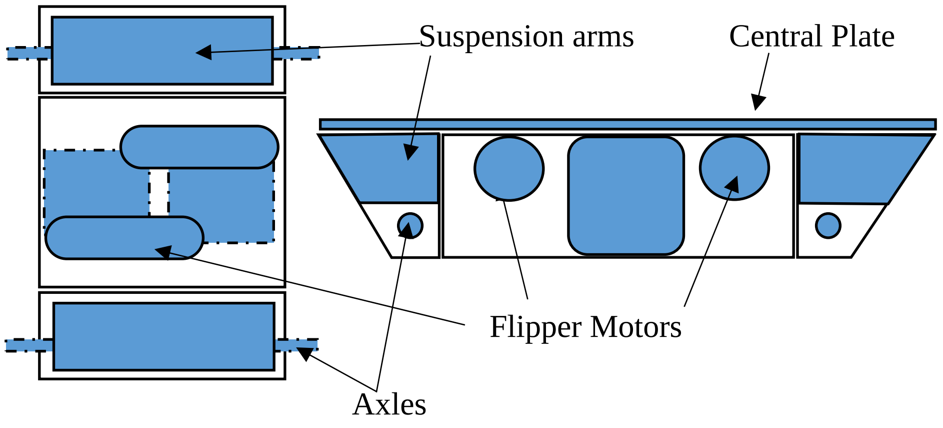


Figure 12: Depiction of component arrangement of the lower section of ATLAS's chassis

Upper Section

The upper section is the main housing for all the electronics, batteries, and control systems, as well as future housing for the arm. Considering the design criterion, the weight distribution of the upper section was determined by the location of the drive motors and the batteries. For simplified construction and manufacture, the drive motors are direct-connected to the drive wheels. Meaning that the motors are positioned at the back of the robot whilst also being high up. The batteries were therefore positioned towards the front, allowing for easy access and counterbalancing the motors at the back. The electronics and arm are then to be positioned in the centre.

The electronic systems were centralised and positioned between the motors and batteries, allowing for electrical connection between all the upper chassis components, as well as positioned above the flipper motors in the lower section (Figure 13).

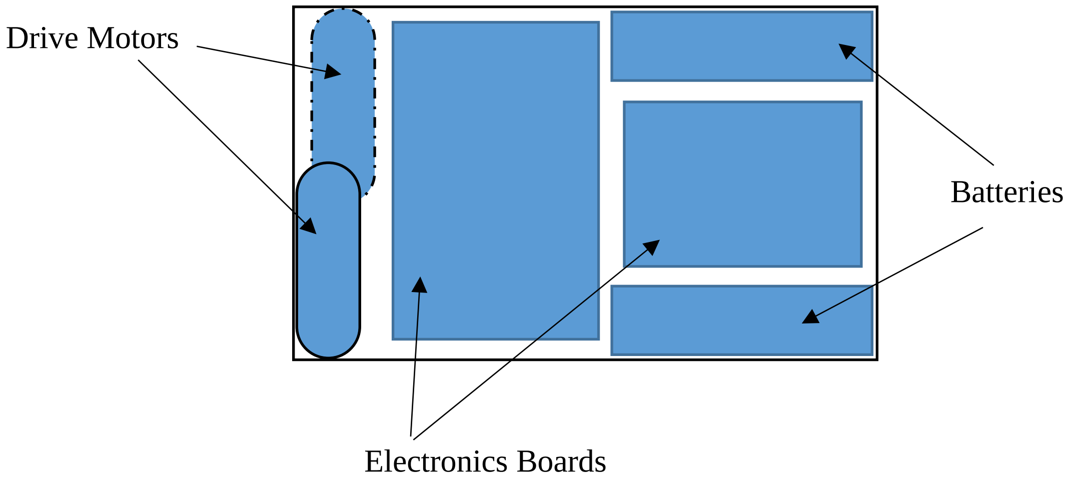
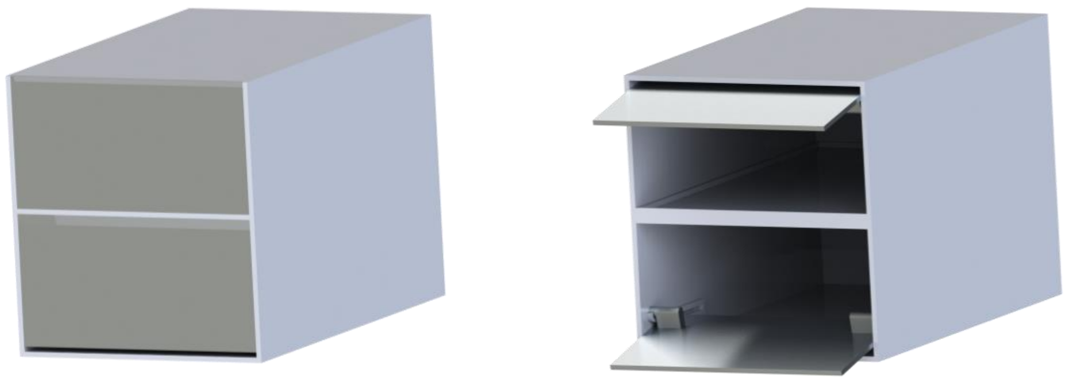


Figure 13: Depiction of the component arrangement in the upper section of ATLAS's chassis

4.4.3. Innovation - Med-Box

As part of maximising the capability of ATLAS within real world scenarios, ATLAS has been equipped with an integrated supply box that can hold med-kits and supplies, such as water and food rations that can be delivered on site should the person be out of reach of personnel help.

The supply box is built into the lower section of the chassis and lies in tandem with the flipper motors. The box opens outwards in-between the main wheels as well as sitting inside the main track. It incorporates a small shelf that separates the box into two sections, rations and med-kit. The doors operate on simple swing hinge and is capable of being slotted into the box when opened to minimise the space in use for potentially cramped spaces.



(a) (b)

Figure 14: CAD image of the Supply Box (a) Closed (b) Open

4.4.4. Innovation – Optimised Chassis

The current chassis of ATLAS features simple hexagonal sections removed from the plates by a water jet. Obviously, this design is not optimised for the loading conditions that ATLAS would experience in the real world, and is significantly heavier than is desirable.

Studies were conducted to investigate ATLAS' ability to withstand typical loading conditions it could experience. It was agreed that ATLAS should be able to survive a fall of 0.5m. The first loading condition considered ATLAS falling on all four of its lowest contact points simultaneously which would be a competition standard test. To replicate a real-world scenario, three more loading conditions were simulated with ATLAS falling directly on its side, on its nose and on one of its corners. These load cases are shown in Figure 15 and their magnitude is calculated in Equations 1 & 2.

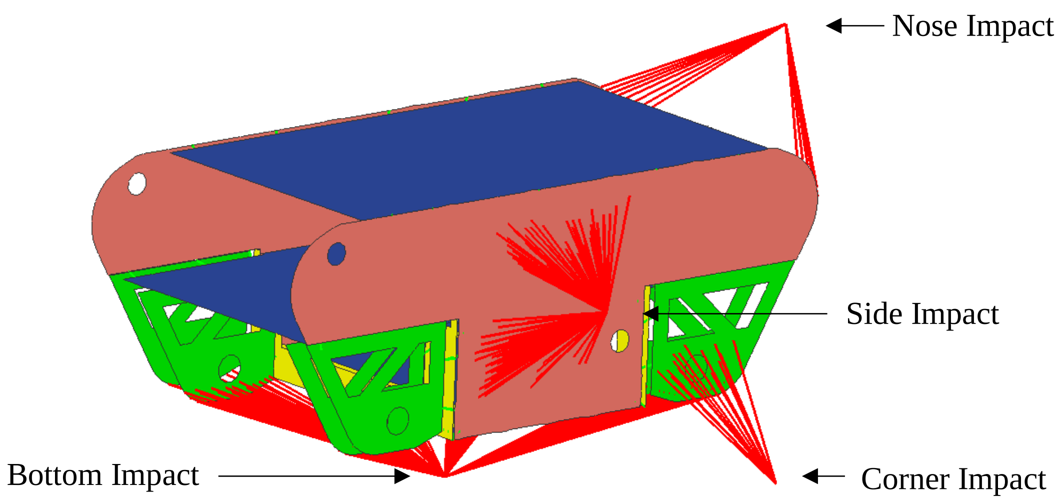


Figure 15: Load Cases for FEA & Topology Optimisation

Using simple mechanical theory and ignoring air resistance, the velocity at impact can be calculated as follows:

$$v_{impact}^2 = u^2 + 2as$$
$$v_{impact} = \sqrt{0^2 + 2 * 9.81 * 0.5} \qquad [1]$$
$$v_{impact} = 3.13ms^{-1}$$

Assuming ATLAS's maximum compression on impact is 0.01m, weighs approximately 40kg unloaded, and that the kinetic energy is converted entirely into elastic potential energy; the impact force can be calculated as given by Equation 2:

$$F = \frac{Net\ work}{Compression\ distance} = \frac{\frac{1}{2}m\,v^2}{d}$$
$$F = \frac{\frac{1}{2} * 40 * 3.13^2}{0.01} \qquad [2]$$
$$F = 19{,}606.9N$$

Bolted connections were simplified to rigid beam element connections for faster calculation and thus faster topology simulation. By doing this however, it assumes the beam is fixed with no deformation and can cause stress concentrations higher than that of the reality of a bolt, which would allow force to translate through the bolt itself.

The simulations were run as linear static analysis using the software Genesis [18]. The target of the study was to optimise strain energy for given load cases and constrained by the amount of material remaining as a fraction of the original. A description of the six different runs that were conducted are as follows:

1. **Baseline Test** – ATLAS' chassis consisting of solid plates with no lightweighting.
2. **Baseline Run with 30% mass fraction** – Testing to see where the critical load paths were forming.
3. **Baseline run with 50% mass fraction** – Test to explore load path priority by forcing more material to remain.
4. **Motor torque load case** – Test to see the effect of the driving motor torques. This was isolated to ensure that one load case does not dominate the results.
5. **Thickness study to 6mm with 30% mass fraction** – Test a future consideration for ATLAS where all plates are 6mm thickness instead of a mix of 6mm and 10mm.
6. **Thickness study to 6mm with 50% mass fraction –** Test for load path priority.

For each variant, data was collected regarding peak stress and maximum deflection for each of the impact cases separately. These results are shown in Table 1. Results in green are improvements upon the previous variant's results and results in orange are worse than the previous variant's results, but still better than the baseline.

Table 1: Peak Stress and Maximum Deflection of Different Load Cases

Variant	**Side Impact**		**Bottom Impact**		**Nose Impact**		**Corner Impact**		**Motor Torque**	
	Stress (MPa)	**Stiffness (mm)**	**Stress (MPa)**	**Stiffness (mm)**	**Stress (MPa)**	**Stiffness (mm)**	**Stress (MPa)**	**Stiffness (mm)**	**Stress (MPa)**	**Stiffness (mm)**
Baseline	89.3	7.6	31.4	0.1	108	3.6	83.3	0.42	N/A	N/A
2	51.6	0.58	10.7	0.013	61.2	0.084	28.6	0.07	N/A	N/A
3	33.6	0.36	7.1	0.015	36.7	0.07	20.8	0.062	N/A	N/A
4	32.6	0.37	7	0.014	36.7	0.069	20.9	0.063	0.0079	1.5E-05
5	43.9	0.68	7.9	0.015	49	0.08	22.5	0.072	0.0095	6E-06
6	38.2	0.49	7.8	0.014	36.3	0.074	21.9	0.067	0.0077	1.5E-05

The yield strength of Aluminium 6082 is 310MPa [19] and with a target factor of safety of 1.5, this allowed for a maximum stress in the model to be 206.7MPa. As can be seen from Table 1, none of the variants were stressed to this extent. A maximum stress of 89.3MPa and maximum deflection of 7.6mm was experienced by the baseline model's side impact. This gives scope for a further investigation to take place to reduce the mass of ATLAS in future years.

The results for each of the load cases for each of the variants can be found in Appendix C. Figure 16 shows the final chassis structure following the sixth variant that exhibits a reduction of mass by 50% from the baseline and features only 6mm plates.

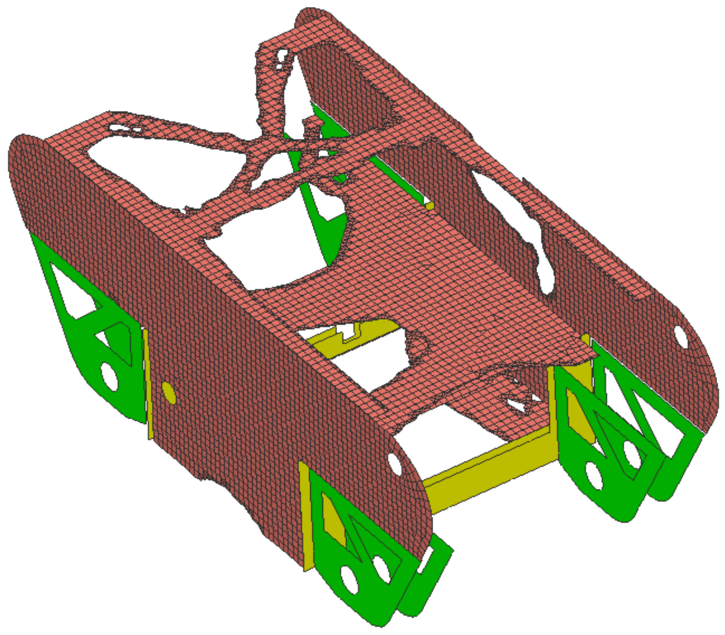


Figure 16: Optimised ATLAS Chassis – 6mm Plates, 50% Mass

From this result, the following conclusions can be made:

- The top and bottom chassis plates are mostly unnecessary when considering the typical loading conditions ATLAS will experience.
- The side plates are critical for the survival of ATLAS in these drop cases.
- The current ATLAS design is over-engineered and has scope for significant mass reduction while maintaining similar performance.

4.5. Suspension Design

A suspension is vehicle subsystem that allows the wheels to move in relative motion to the main body of the vehicle; its main purpose is to improve ground contact by enabling the wheels to follow uneven ground profiles. It must also mitigate and eliminate any shocks or impacts that occur because of rapid change in the ground profile. It therefore allows a machine to move at higher speeds with suspension than it would without. In the case of ATLAS, the core aim is not to improve speed, but to reduce the level of impacts and shock induced by falling and bouncing.

The decision to design a suspension system is a result of forensic analysis of Champion, which revealed that the system suffered excessive damage to key components as shown in Section 4.2.

Therefore, there is a requirement to attempt to minimise the damage done. This could be done by strengthening the chassis with the addition of material and plates to stiffen the areas that may come under effect of the shock; this does however induce greater mass and size as a by-product. Although more complex and potentially expensive in the short term, a suspension is more suitable in terms of space and size, whilst providing a long-term solution to preventing component damage.

4.5.1. Critical Review of Previous Designs

The suspension style adopted by Orion was a vertical spring suspension (VSS), the design was capable of 12mm travel; however problems created by interaction of components resulted in travel distance being severely inhibited. The mounting positions of the spring dampers was also of questionable position, with the mounting blocks fixed in a parallel axis to the movement of the springs and the force acting upon the suspension. This created potential mechanical failure of the suspension blocks slipping under heavy load.

Cyclone's suspension method was based upon the concept of torsion bar/blade suspension (*Figure 17*); the torsion bar element was laid underneath Cyclone's chassis and connected to the rollers at both ends. A fundamental issue with the design however was that the torsion blade was not one solid component, but connected to the wheel sets via a component dubbed the 'torsion blade adapter'. It was found that the blade adapter created a point of weakness, due its material choice of a polymer. As such, the adapter was brittle and snapped when subject to load.

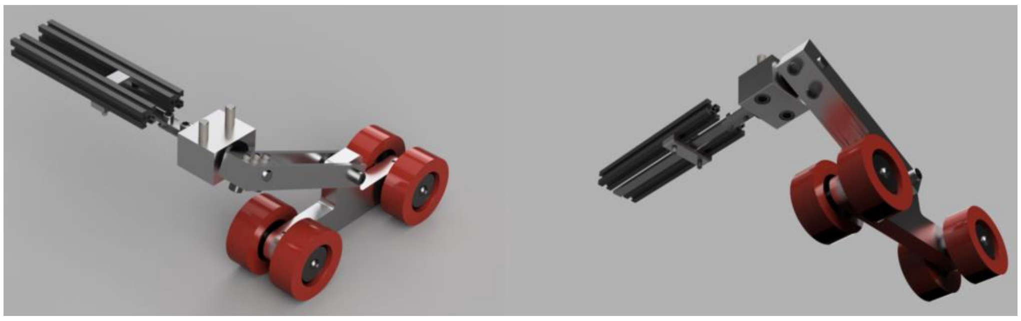


Figure 17: Cyclone's Torsion Bar Suspension System

Hydropneumatic suspension

- A hydropneumatic suspension is composed of a hydraulic cylinder, fluid and hydropneumatic accumulator. It operates by displacement of the hydraulic rod to change the fluid volume to create a change in pressure, the change in pressure alters the force on the piston rod [20].

- Hydropneumatic suspension has a very high efficiency and effectiveness; in terms of space the components it can be small and spaced in regions away from the suspension zone should packaging be an issue [20]. Its high capability make it the highly favoured suspension used in modern tanks such as the British Challenger II.
- Hydropneumatic suspension is a more difficult and costly system in comparison to spring based suspension systems. This is due to the requirement for precise level control of the hydraulic fluid, incorrect measurements and levels invalidate the use of the suspension [20].

Vertical Volute Spring suspension (VVSS)

- Vertical Volute Spring Suspension (VVSS) uses a coil spring that is connected to a pair of swing arms that hold the wheels. Under compression, the wheels slip in a lateral direction while the spring-damper resists the compression. (Figure 18)
- As a system, the VVSS is a relatively simple, tried and tested concept, being used in tanks back in the mid-1900. Its spatial requirement is very low by placing all components within a single plane of operation and is very resistant to overload [21].
- Due to large stresses created within the spring, the lifespan of the suspension is reduced; it also requires a suitable fixing position that will not yield to bending moments acting perpendicular to the orientation of the attachment point [21].

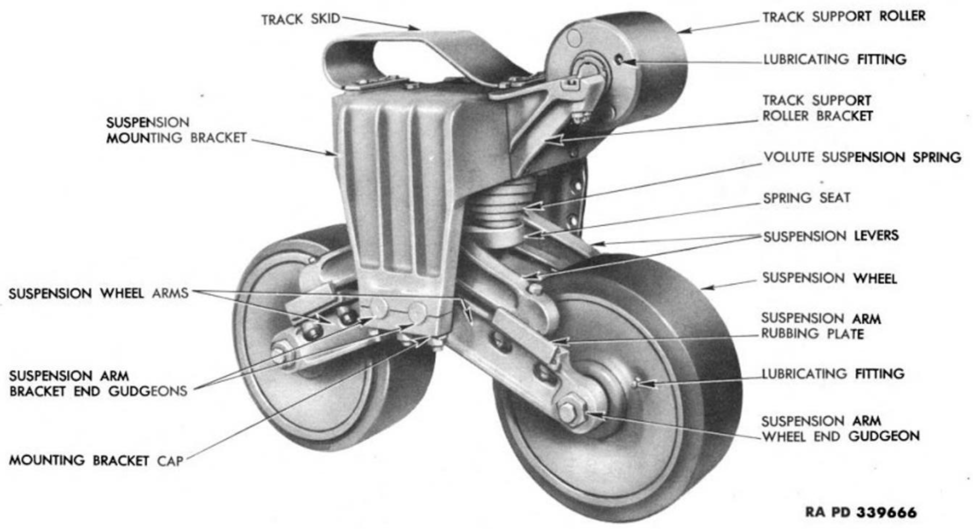


Figure 18: Vertical Volute Spring Suspension [21]

Torsion Bar

- A torsion bar suspension is constructed of a long bar that is connected to a wheeled swing arm. The bar is preloaded to create a minimum ride height. When the wheel has force loaded on it, the bar is placed in torsion, utilising the natural stiffness of the material to resist torsional movements as a spring.

- The torsion bar suspension is a simple concept to implement as well as being relatively cheap to manufacture. It is also a durable system and possesses a small area profile, however is does require a large length across the breadth of the robot, potentially inhibiting lower situated components.
- The torsion suspension is not particularly adaptable, due to the stiffness being determined by the cross-sectional moment of area, length and material property. The simplest method to alter the suspension to fit different terrains is to change the torsion bar itself to change the spring stiffness. This requires extensive manufacturing time just for adjustments.

4.5.2. ATLAS - Solid Suspension Design & Manufactured Parts

As part of the design process it was determined that a solid suspension would be designed for the initial iteration and implementation before attempting dynamic suspension. For maximum strength and stiffness, the suspension system is integrated into both the upper section and the lower section, utilising the concept of modular arm attachments (Figure 19). The suspension is thus connected in a secure manner that can withstand the required loads of the robot. As part of the adaptive development process, the suspension is modular to allow for the development of simplified functional components into further, more complicated and optimised variants.

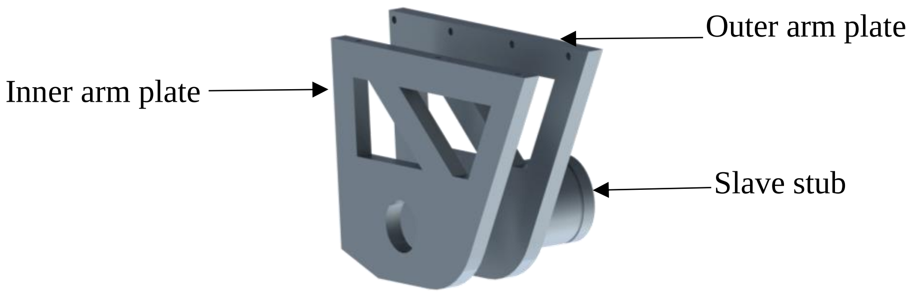


Figure 19: Modular suspension arm (Solid Suspension Variant)

In this case, the arms are replaceable, first being manufactured to complete the role of utilising a solid suspension. This system is relatively simple, mainly focusing on rigidity and strength to absorb impact. Thus, the arm for the solid suspension is simply a series of 10mm thick aluminium sheets that are directly attached to the chassis and the stubs that the flipper axles sit within. Therefore, the total functionality of the robot must be modified. When using the solid suspension, priority must be given to strengthening sections that will take load and minimise the quantity of components that are susceptible to bending and torsion.

In this fashion, the method of driving the lower flipper arms must take on a flexible; yet rigid approach. Rather than the more effective worm drive, a chain drive is being implemented. This chain enables a direct connection between the motor shaft and the axle that the flippers rotate upon, thereby reducing the number of shafts and components that are exerted upon during impact. The major drawback of this design is that it exerts a large bending moment on the motor shaft when the flipper arms are under load. Later iterations utilising dynamic suspension and subsequently a worm drive would remove this potentially fatal characteristic.

The axle itself is comprised of three sections, a central steel bar that possesses a keyway, allowing the driving cog to be attached. Two aluminium rods are connected on both sides, changing the radius of the material to match that of the re-used axle stubs that the wheels will sit upon. The purpose of a thinner central bar is to maximise strength and minimise weight, whilst still adhering to the use of older components from previous years (Figure 20).

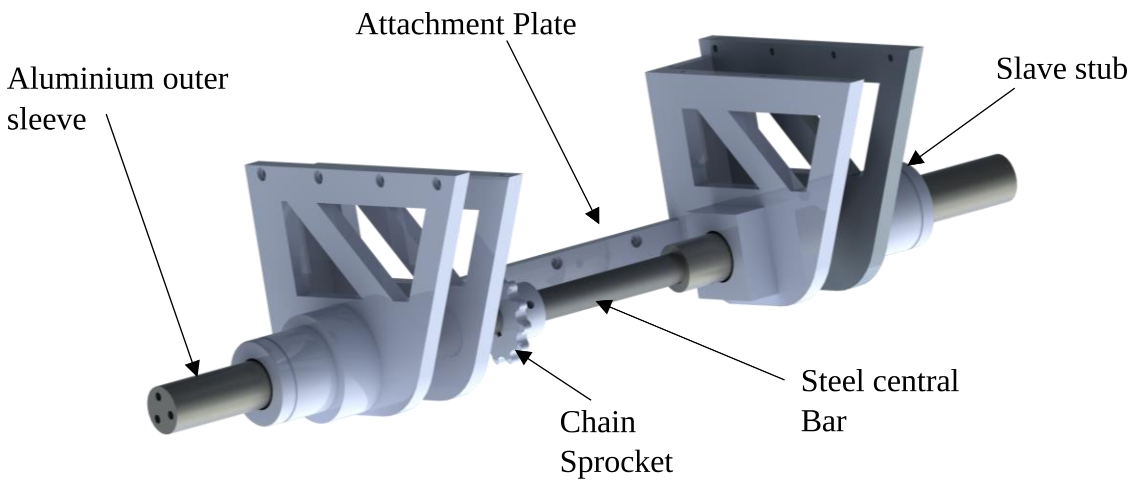


Figure 20: Assembled solid suspension in conjunction with axle and attachment bracket

4.5.3. Innovation - Dynamic Suspension Design & Calculations

Using the solid suspension as a base, the eventual aim is to move towards a dynamic suspension that can change in accordance with the terrain. With this change, a large part of the lower section of the chassis must adapt with it, however, the fundamental shape of the chassis will remain the same. The major feature will simply be the exchange of different panels of aluminium to better accommodate the modifications.

After consideration of several designs, the main outline of the dynamic suspension will incorporate a swing-arm spring-dampener combination that is commonly seen on motorbikes. The design is a tried and tested concept that possesses a high flexibility, coupled with a relatively simple, compact and effective design (Figure 21).

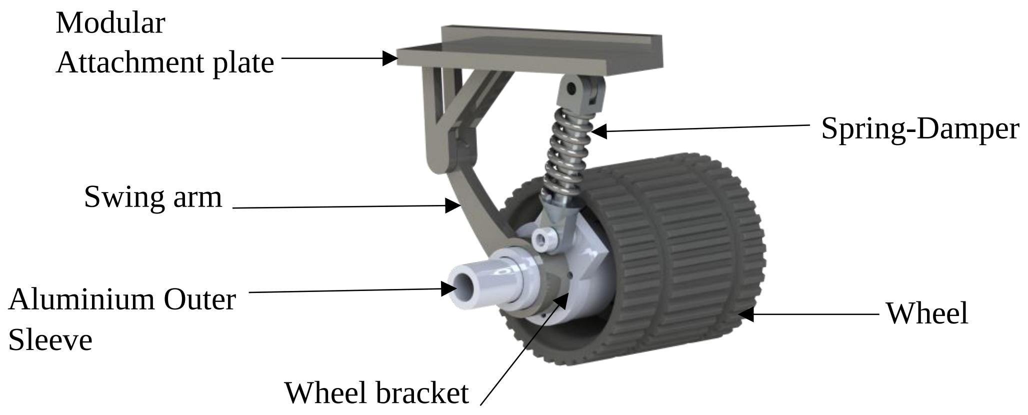


Figure 21: Dynamic Suspension System Design

ATLAS's design would have a pair of swing arms that are connected directly to the chassis in place of the current solid arms, utilising the same connections points as part of the intended modularity. The swing arm is subsequently connected to a bracket that is composed of the slave stub and spring-dampers that connect back to the main chassis. The effectiveness of the design is based upon its capacity to limit the movement of the suspension to only one plane, whilst allowing the wheels to spin freely. At the same time, this allows the flipper arms to be connected in similar manner to the solid suspension system; in that they are being driven by a single axle.

However, due to the new capability of the stubs moving within a plane, the axle also must be capable of movement, whilst also allowing the two wheels to move independently of each other. This requirement renders the solid axle as null, due to its high potential for causing the suspension to lock and become rigid. The simplest solution is to create a split axle that is reminiscent of a car axle, using a combination of universal joints and spring-damper bars that can move in 3-D space. In a similar manner to the original axle, the new dynamic axle will also be separated in to three components, with a central section and two extensions on either side.

With impact forces now being absorbed by the suspension, the requirements of the flipper drive system a more relaxed, allowing the axle to be driven utilising a worm gear setup. This in turn means that the orientation of the motors can be modified, given that the original positioning was done to enable the use of a chain drive. The motors can now be moved to have the shafts run parallel with the chassis of ATLAS and direct-connected to the newly elevated dynamic axle, which is held in place using a pair of pillow bearings. The overall system is now adapted for dynamic suspension with changes done to the central plate fixings and the suspension arms.

4.6. Tensioning Design

Tensioning systems on tracked vehicles such as ATLAS are required to maintain tension on the tracks while the vehicle is traversing difficult terrain. They can be either static or dynamic, depending on whether the track is pre-tensioned before operation or tensioned dynamically as the robot moves respectively.

4.6.1. Critical Review of Previous Designs

Orion and Cyclone both featured similar designs for their dynamic tensioning system as seen in Figure 22. Their system featured two springs acting against a pillow block, which applied a force from within the chassis to an axle connected to the idler wheel on the exterior.However, this design had a glaring flaw, which was revealed once it was manufactured and tested. As the pillow block assembly was situated inside the chassis, a moment was induced between the block and the idler wheel. This moment caused rotational movement and the springs jammed, failing to compress further and ultimately led to the tracks falling off the wheels.

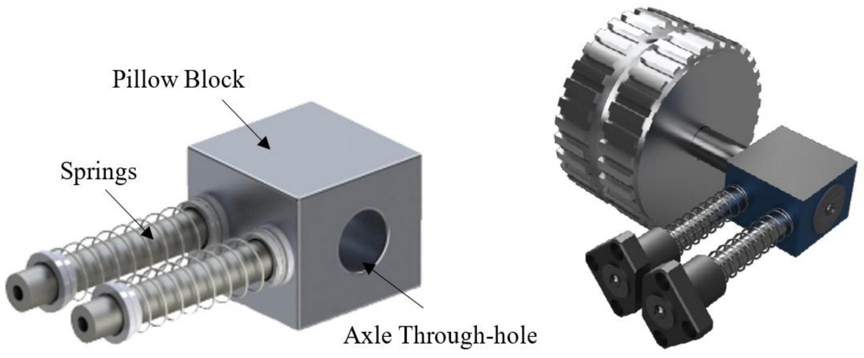


Figure 22: Previous iterations of dynamic tensioning designed by previous WMR teams

4.6.2. ATLAS - Static Tensioning Design & Manufactured Parts

Improving upon the design of the previous years and in keeping with the strategy employed when designing ATLAS, it was decided that a static tensioning system should first be implemented.

Figure 23 shows the chosen design. With respect to the limitations of the previous designs and considering the extra space generated by ATLAS' larger chassis, it was deemed possible to remove the moment issue. This was achieved by detaching the axle from the main chassis and having a separate system that supports the wheel in place directly in its centre.

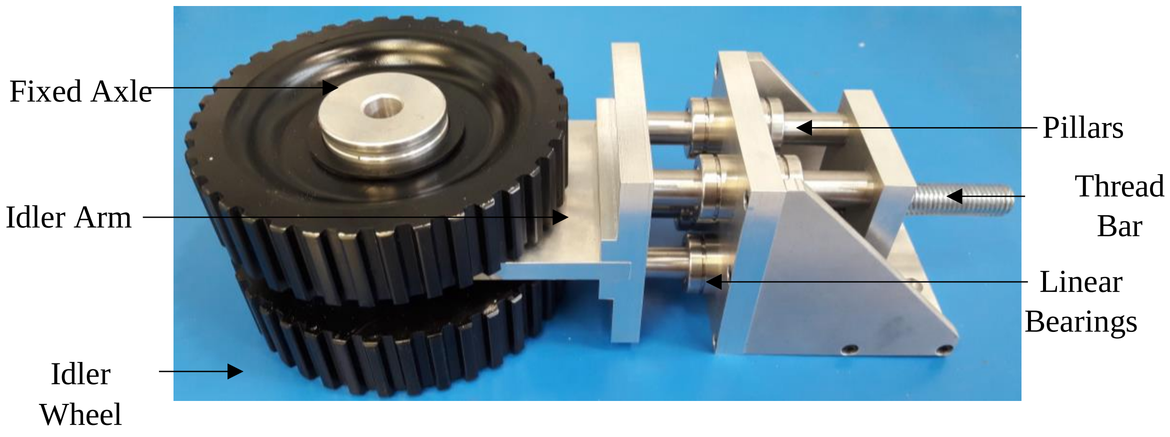


Figure 23: Fully assembled static tensioning system

The system's independence of the main chassis allowed it to be mounted on the side of the frame at the largest allowable angle of 30 degrees (Figure 24). Four pillars were chosen for robustness and these slide along linear bearings to ensure smooth operation. In the centre of the pillars is an M12 threaded rod that can be tightened with an easy-to-access double nut combination. This enables the track to be slackened or tightened -20mm to 40mm respectively from its 'neutral position'.

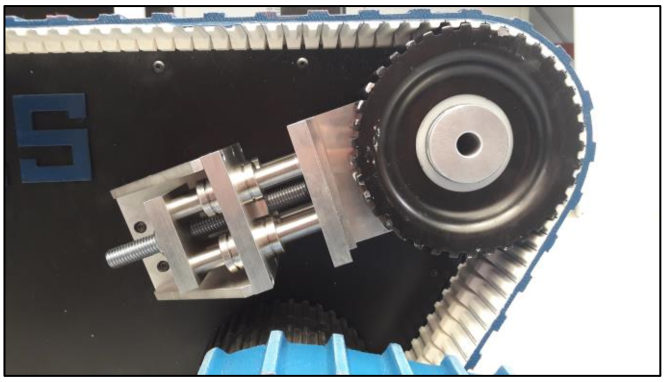


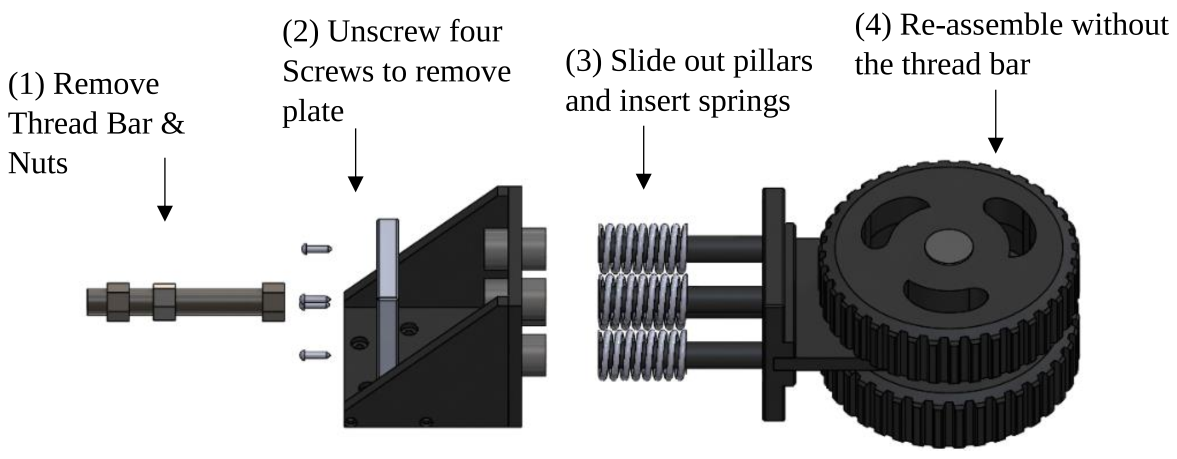
Figure 24: Static Tensioning system upon attachment to the chassis

There is a flaw to this design that has been identified and that is its mass. The static systems weigh 2.5kg each and are situated in an elevated position on ATLAS, moving the centre of mass upwards and forwards, which is not ideal. A potential innovation for the future can be to reduce the mass of the system or find a method of increasing its angular position.

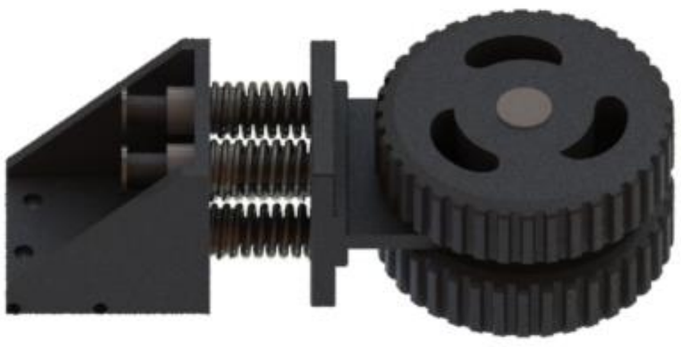
4.6.3. Innovation - Dynamic Tensioning Design

For the future of ATLAS, a dynamic tensioning system has been proposed which would work in tandem with the suspension system. The dynamic tensioning is an adapted version of ATLAS's static tensioning system that rids itself of the threaded bar assembly and instead opts for four springs mounted atop the linear bearings.

With ATLAS’ strategy of simplicity in mind, should the dynamic tensioning system fail in service due to unforeseen circumstances, the system can be returned to static tensioning without removing it from the chassis. The method of changing from static to dynamic and vice versa is shown in Figure 25a. The full dynamic tensioning assembly is shown in Figure 25b.


(1) Remove Thread Bar & Nuts
(2) Unscrew four Screws to remove plate
(3) Slide out pillars and insert springs
(4) Re-assemble without the thread bar

(a)



(b)

Figure 25: (a) Assembly method converting the static tensioning to the dynamic tensioning, (b) Fully assembled dynamic tensioning

Springs are the simplest way to provide the force necessary for counteracting the slack experienced by the tracks, but the selection of the correct spring rates is crucial. This system benefits from having the load distributed across four pillars, allowing for lower spring rates.

Two major design elements were identified for the dynamic tensioning system’s springs that need to be considered in future years:

(1) The springs must already be in compression whilst the track is in its desired ‘neutral’ position. This is to compensate for the lack of static tensioning capability.

(2) There must be sufficient space for the spring to oscillate about this position, to account for varying profiles of the terrain. This means the dimensions of the spring and its maximum allowable compression become significant.

Due to the complex nature of the system in place, too many assumptions would have to be made for accurate hand-calculations for spring-rate. It is therefore recommended that future years use computerised methods for determining the spring-rates in conjunction with suspension design.

4.7. Flipper Arm Design

Flipper arms are used as extensions of the main track to give extra traction when climbing steep, difficult terrain. This section describes the evolution of the flipper arms on WMR robots and ATLAS.

4.7.1. Critical Review of Previous Designs

Between 2008 and 2013, Warwick's Search and Rescue Robot used the same principle of flipper arm design every year. The majority of the design was reused and only minor alterations were made to the geometry in certain years. Each year the team reasoned that the flipper had performed adequately and that it was desirable to reuse parts from the previous year to reduce their own design and manufacturing times. The 2013 flipper assembly is shown below in Figure 26.

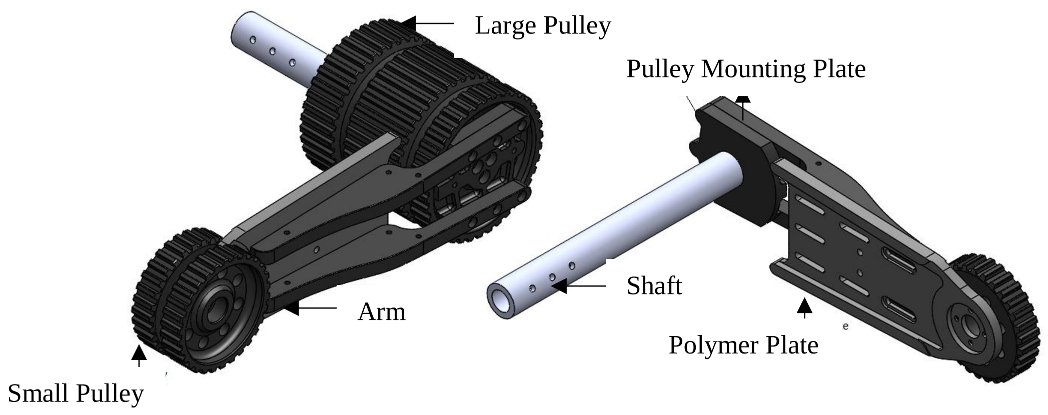


Figure 26: Diagram of the 2013 flipper design

On physical inspection of the 2013 flipper, several flaws were observed. The geometry meant that any bending moments about the flipper's length would solely be resisted by the material strength at the thinnest sections and the resistance of the bolts to shearing. The polymer plate was made from Nylon 66 and was at some points only 4mm in thickness. The flipper showed considerable deflection when twisted by hand. Such deflection is a significant concern because it becomes possible for the flipper tracks to become detached from their pulleys.

There was also noted to be considerable loose yawing motion of the flipper about its length. This was due to the flipper assembly only being attached to its shaft at a very small area of contact by four bolts in the middle of the shaft. The pulley mounting plate does appear to fit inside the large pulley; however, there is not a continuous contact between the two. Since the pulley mounting plate is not fully constrained, this also contributes to the yawing motion. This yawing of the flipper is undesirable because it can hinder the robot's handling and reduce the traction applied to the ground.

The 2013 flipper does have the ability to statically tension its track, but this is not a fast or convenient system as six bolts need to be unscrewed then reapplied.

Previous teams had also never tested the flipper assembly in a drop test simulation, as this had always been performed only on the main chassis section. Some environments, such as the step field, could require the robot to overcome obstacles with all flippers inclined down, before it experiences a small drop on one side.

Due to all these issues, it was decided to develop a new flipper design for ATLAS. The objectives for the new design to meet included:

- Increased stiffness and reduced deflection.
- Include a quick and simpler static tensioning system.
- Withstand drop impact force from a height of 0.15m.
- Ability to easily vary its length to allow future teams to use a longer flipper arm.

4.7.2. ATLAS - Flipper Arm Design

A slider mechanism was proposed to allow the flipper assembly to vary its length to tension the track. The slider supports the flipper's smaller pulley and the slider is housed inside the holder that supports the flipper's large pulley. The slider and holder will be made from Aluminium 6082 and the holder will ideally be made as one solid piece to include the former arms and mounting plate. The extension of the slider is controlled by an M6 bolt. This allows easy tensioning of the track but also allows the track to be removed quickly if needed. On the lower side of the flipper, a sledge has been incorporated to reduce the stresses on the track.

The flipper assembly would no longer attach to the shaft using bolts through the shaft's cross section. Instead, the end of the shaft has its cross section cut as a hexagon shape that fits inside a hexagon hole in the holder. The flipper shaft is also modified to incorporate a flat plate that is bolted directly to the holder. These two features can eliminate the yawing motion.

The flipper is arranged so that the contact area between the slider and holder provides extra resistance to both bending and twisting, and so significantly stiffens the assembly. This design would no longer rely on the resistance of bolts to shearing forces. This design was rapid prototyped and is shown in Figure 27.

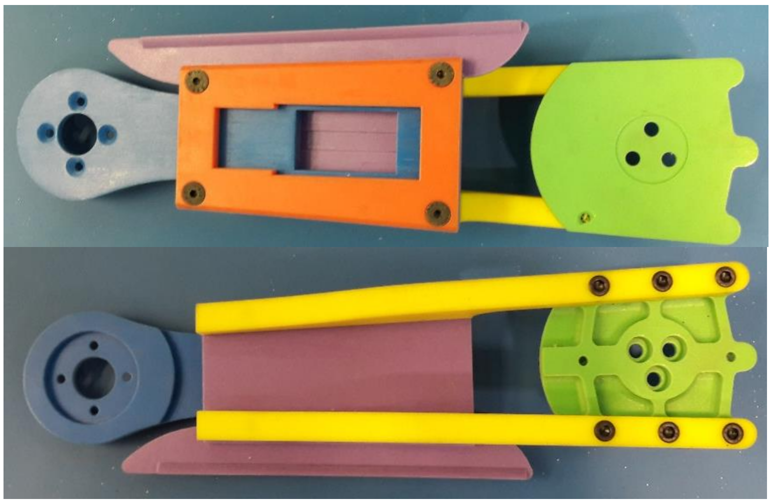


Figure 27: 3D printed model of ATLAS's flipper design

4.7.3. Extendable Flipper

The proposed design could not be fully followed due to manufacturing constraints. Creating the holder as one single part would be an unfeasibly expensive process. The final design would incorporate the proposed principle but would now manufacture the holder from two pieces. The flipper shaft's attaching plate and hexagon end-cut were also omitted due to concerns it could create difficulties during assembly. The design allows the length of the track to vary by 40mm. The slider can extend further, but becomes no longer firmly clamped by the holder. This would cause excessive deflection of the track and because of this; the flipper's length was limited to vary by 40mm. The large and small pulleys from the 2013 flipper will again be used. This proposed flipper system is shown in Figure 28.

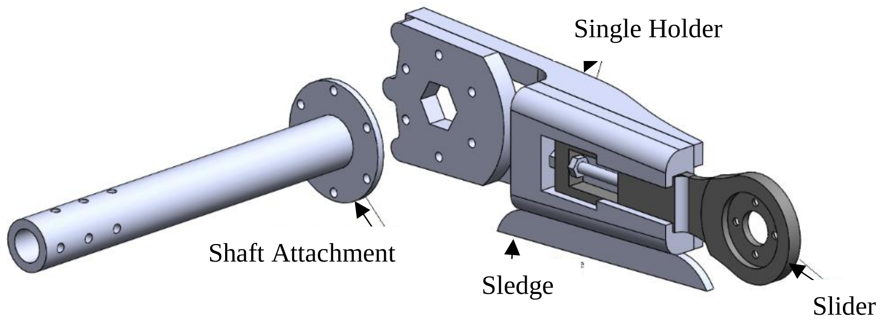


Figure 28*:* Innovated Flipper Design

4.8. Mobility Calculations and Centre of Mass

ATLAS was designed to keep its centre of gravity (COG) as low as possible to improve its manoeuvrability. ATLAS has been designed using a lot of symmetry based around a central trapezoid chassis and this makes its 52kg mass evenly distributed where the COG almost coincides exactly with the vehicle's geometric centre. This gives ATLAS similar manoeuvrability when travelling in forward or reverse directions. The relatively low COG is beneficial because it allows steeper gradients to be climbed and means that any high acceleration will result in a lower amount of weight transfer occurring in the chassis.

The COG occurs at a height of 146mm above ground. When viewed head on, the COG is only 2mm away from the midpoint of the vehicles track. When viewed sideways, the COG is only 17mm behind the midpoint of the vehicles wheelbase. This is shown below.

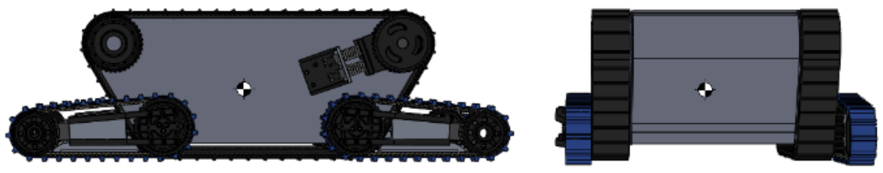


Figure 29: Image depicting ATLAS's Centre of Mass

ATLAS's low COG and wide footprint allow it to traverse steep inclines without toppling. In a theoretical situation of unlimited grip where the tracks do not slip, ATLAS can traverse inclines whilst its COG is positioned directly above the outermost edge of its track in contact with the ground. This means that ATLAS is capable of climbing a 70^o incline while travelling forward at constant speed. ATLAS is also capable of travelling in a straight line across a surface, which is inclined at 60^o to the horizontal.

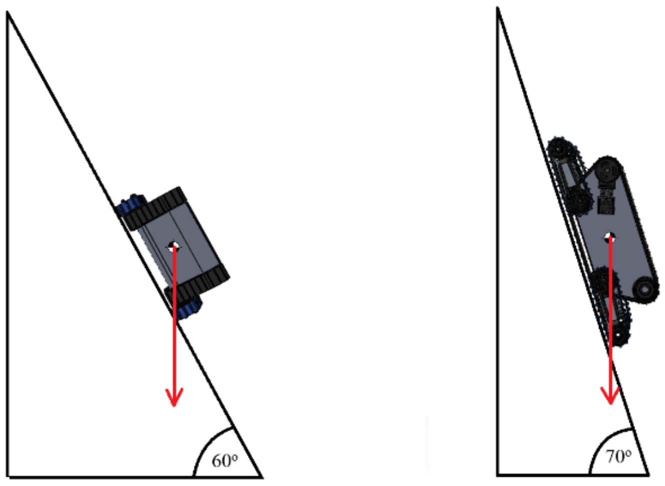


Figure 30*:* Image depicting ATLAS's maximum possible climbing capability at different orientations

The tractive force that a track exerts upon the ground is dependent on both the normal reaction force at the track and the coefficient of friction, μ, between the track and the ground. However, the value of μ will vary as the vehicle traverses over different materials in different conditions that may be wet or dusty. Because ATLAS will be expected to traverse all such terrains, it is not possible to consider all these in one calculation and so friction will be ignored. These calculations represent the best possible scenario where the tracks outer teeth are engaging and locking with indentations in the ground as the track rolls without slipping.

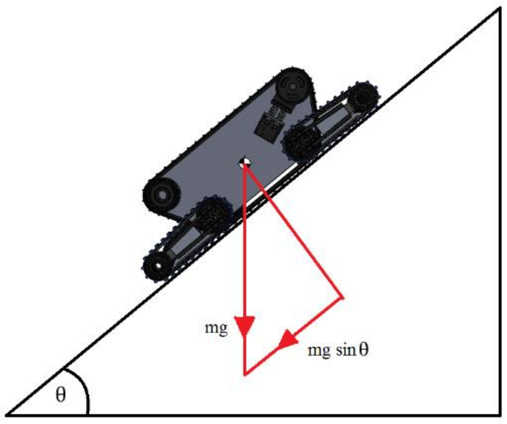


Figure 31*:* Forces acting upon ATLAS during climbing at an angle θ

The simplest model can be used to find the torque necessary to hold ATLAS stationary on an incline or climb the incline at a constant velocity. In such cases, the tractive force produced by both motors is equal to the component of ATLAS's weight parallel to the incline. In this model, the robot's rolling resistance is neglected, as this needs to be determined from physical testing.

The torque required from each of the motors can be expressed as:

$$\tau = \frac{\text{mg}\sin\theta\text{ r}}{G\,n}\left(\frac{100}{e}\right) \qquad [3]$$

Where m = the mass of the robot, r = the radius of the annulus within the driven pulley. G = the overall gear ratio between the motors and the driven pulley. N = the number of motors driving the tracks and e = the estimated efficiency throughout the motor and the drivetrain.

The motors were originally purchased and used by the 2016 team and hence the efficiency of the motors are estimated to have fallen to 75%. The gearhead has a listed efficiency of 83%. This gearhead spindle drives an annulus within the driven pulley, which was manufactured without involute teeth; this gear pair has an estimated efficiency of 80%. This gives an overall efficiency of 50%. The gearhead has a gear ratio of 26:1, while the gearhead spindle and annulus pair have a gear ratio of 6:1. This gives an overall gear ratio of 156. In the case of traveling at constant speed up a 45 degree incline, the required torque is:

$$\tau = \frac{(52)(9.81)(0.707)(0.04)}{(156)(2)(0.5)} = 0.092\text{ Nm.} \qquad [4]$$

The motors have a maximum continuous torque of 0.405 Nm. Even by being programmed to run at up to 90% of this value, the motors provide more than the necessary torque and are capable of accelerating the robot up the 45 degree incline instead of simply holding itself stationary.

In reality, each motor provides 708 N of tractive force through each of the main tracks. If the motors were connected to an external belt, then the motors would be capable of lifting the robot up an incline of 90 degrees.

4.9. Mechanical Summary

Previous WMR teams had issues with mechanical failures due to complex designs. ATLAS' mechanical systems consist of a simple chassis structure with rigid suspension, a static tensioning system to maintain track tension and redesigned flipper arms. For the future of ATLAS, an optimised suspension topology, dynamic suspension, dynamic tensioning and improved flipper arms have been designed and proposed.

5. Electronic & Software Design

Having established the mechanical design of ATLAS, Section 5 gives a detailed overview of the electronic and software systems implemented in to the 2016/17 robot.

5.1. Systems Overview

The electronic architecture of ATLAS may be broken down into two main subsections; the base station and the ATLAS station. The base station consists of a master laptop, router and PlayStation 3 (PS3) controller. The ATLAS station consists of the electronic hardware on-board the robot, culminating with the Pico computer, which acts as the main processor. The Pico communicates with the master laptop via the on-board and off-board Wi-Fi routers.

Overcurrent, as a result of high stall currents from the motors, can result in the burnout of the electronic components and so as a precaution, current sensors were integrated. Furthermore, a voltage sensor in the form of a battery monitor was implemented to avoid battery degradation caused primarily by extensive discharge. An additional array of sensors was incorporated to fulfil the RoboCup specification.

A systems diagram detailing all the components may be seen below in Figure 32.

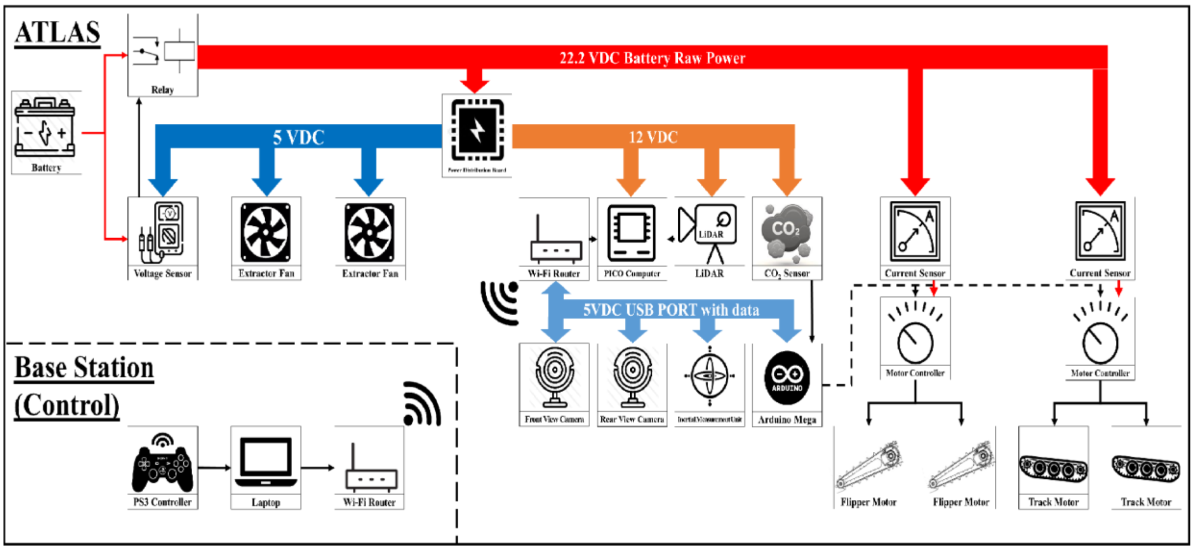


Figure 32: Systems Diagram

5.2. Electronics and Software Literature Review

Power distribution boards (PDBs) are used extensively throughout electronic systems as a means of dividing electrical power from the supply system to subsidiary systems. There are various methods by which one can regulate the voltage and thus distribute the required power throughout a system. Common methods of power distribution utilise voltage regulation. Voltage regulation is required to create a voltage reference from which the subsidiary circuit can operate at a stable voltage [22]. Lithium Polymer (LiPo) batteries are known for their high energy stores relative to their weight which is the primary reason they have been utilised for this project. It is known that they are inherently unstable and consequently produce a ‘noisy’ output waveform [23]. It is therefore the function of a voltage regulator to create a smooth and stable voltage reference. This is primarily achieved by reducing the ripple of the output.

The most simple of these voltage reference topologies includes Zener diodes. However, a significant drawback is that both the diode and resistor require high power ratings to handle the current draw. This becomes impractical at high voltages and current. Consequently, Zener diodes are typically used up to 5W [24]. In high voltage or current scenarios it is more practical to use integrated circuit (IC) regulators. *Horowitz et al.* states strongly that *"the user of such supplies should not try to design and build them – buy them from the expert folk who do this for a living"* [22]. IC regulators not only provide stable voltage reference but offer improved safety, step-up capability, multiple outputs and ground loop prevention [25].

Analysing the evolution of WMR power distributions systems will allow the team to recognise drawbacks from real-world circuits and how these were overcome. Identifying these characteristics and using the plethora of past information will allow the team to continuously improve. It was first identified by the 2012/13 WMR team that the power distribution, especially within the head of the arm was *"haphazard, illogical and confusing"* [26]. It was reported that the 2011/12 project team experienced both intermittent communications and power dropouts. Further to a re-wiring of the system, the team re-defined connection ports to help define the variety of voltages and polarities accumulated over the years. This was achieved via simple marking on the board [26].

The 2014/15 project team, named Orion, identified that previous teams had used *"poorly organised"* and *"large stacks of dated computation boards"* [27]. It was reported that boards had under-gone multiple design iterations to account for the multiple output voltages required over the years of projects. Furthermore, to reduce the size of boards, due to the volume constraints within the robot, and remove the need for multiple switchable outputs it was decided to use IC regulators. This is a key feature that the current WMR team intend on utilising. However, the intermittent connectivity was still reported to be a problem [27].

The 2015/16 WMR team, named Cyclone, aimed to resolve the still present connectivity issues. They attempted to resolve via the addition of second router at the base station. Preliminary testing was performed, although the system was not fully developed. The addition of the second router did benefit the system, as it allowed Cyclone to operate within a greater range, up to 13.5m [27]. The routers placed within the shell of the robot experienced approximately 50% attenuation [27]. The ATLAS team intends on utilising this significant improvement. Another important improvement developed by the Cyclone team was the removal of the 8-to-1 USB splitter. Proven to be a bottle within the system. It was proposed that RS232 ports on the robot computer were utilised, offering improved data transmission speeds for both the CO_2 sensors and the IMU, improving video streaming quality [27], again this technology has been utilised within the ATLAS project. It was found that both the Orion and Cyclone team made assumptions and not accounted appropriately for component tolerances and an adequate safety factor. It was recommended that that power requirement were not taken from the respective datasheets but experimental data is also acquired. This is something the ATLAS have taken into consideration when designing electronic schematics. TRACO TEN60-2412N and TRACO TEN40-2411N, IC regulators were bought by the Cyclone team to manage their new power distribution system [27]. Harwin connectors were also bought to aid with the identification of voltage ports and polarities.

The ATLAS team intend to utilise the many improvements identified throughout the past 5 years of WMR projects. Capitalising on the abundance of experimental and real-world testing available. Recognising how and where difficulties were overcome and utilising this within ATLAS's project.

Following the team's decision to adopt a modular approach, the requirements for the power electronics board was split into a Power Distribution board, Battery Monitoring Circuit and an Overcurrent Protection Circuit. This would allow for individual testing of each module, as well as improving adaptability for future years; preventing the need to disassemble the entirety of the power system.

5.3. Power Distribution

5.3.1. Critical Review of Previous Design

A review of the 2015/16 power distribution established that the manufactured boards provided regulated power outputs of 12V and 5V, whilst also including current protection. However, it was found that the choice of connectors in regards to the output terminals meant that connecting extra components was challenging. It was also discovered that the limited number of terminals included were not sufficient to power all required subsystems necessary for the 2016/17 team.

Therefore, considering the limitation of the 2015/16 board and accounting for the application of the robot arms in the future, it was decided that a new Power Distribution Board should be designed.

5.3.2. ATLAS – Power Distribution Board Design and Manufacture

ATLAS' power requirements for each component were recorded and listed in the following table. To account for any over-current and component tolerances, a 20% safety margin was added.

Table 2: Total Power drawn from the Power Distribution Board

Power Source	Device	Voltage (V)	Current (A)	Safety Margin (+20%) (A)	Power with Safety Margin (W)
TEN 60-2412N	LiDAR	12	0.7	0.84	10.08
	Pico Computer	12	1.5	1.8	21.6
	CO2 Sensor	12	0.5	0.6	7.2
	Inertial Measuring Unit	5	0.5	0.6	3
	Arduino Mega	5	0.5	0.6	3
	Front View Camera	5	0.5	0.6	3
	Rear View Camera	5	0.5	0.6	3
TEN 40-2411N	Extractor Fans	5	1	1.2	6
	Current Monitor	5	2.5	3	15
				Total Power drawn:	**71.88**

All the electronic devices operate at either 12V or 5V. The total power required from the 12V supply is 50.88W and 21W at 5V. It was identified that two separate Direct Current (DC) isolated power supplies were required to step down the input voltage from the battery at 22.2V to the required two voltages. Once the power and voltage requirements of the DC isolated power supplies had been identified, it was established that the TRACO TEN60-2412N and TRACO TEN40-2411N purchased by previous WMR teams fit the needs of ATLAS' power board. The two DC isolated power supplies were initially tested to check they were operational and consequently selected to be reused. Not only do the TRACO power supplies have an excellent efficiency of up to 92% but they can be easily mounted on a printed circuit board (PCB).

To prevent damage to the components from overcurrent that may be caused by power surges, a fuse is required to protect the circuit [28]. The total power drawn from the Power Distribution Board at 22.2V is 71.88W. The current draw will be 3.2A. Considering power surges at start-up, a fuse rated at 5A is necessary to allow current to flow; acting as a safe-switch when over-current occurs.

In the design of the Power Distribution Board, resistors are configured to create a voltage divider. When combined with a slide switch this acts as a control mechanism for the two DC Converters. When the switch is ON, it will connect both resistors to the Remote Pin of the DC Converters. This is to provide a voltage between 3 to 12 VDC that will enable the voltage conversion to begin. When the switch is OFF, the Remote Pin will be connected to the negative branch of the circuit, disabling the converter. The schematic and PCB design can be seen in Appendix F.

5.3.3. Innovation - Increased Terminals and Simplified Connectivity

The addition of new output terminals increases the available output ports to allow new components to be added. Screw terminals are utilised to provide a simple means of connecting to the board and removing the need for specific male to female connector pairings. Furthermore, the spare capacity of the DC converters used will allow for future team to be able to add new components without needing to redesign a new board.

5.4. Battery Monitoring

Lithium Polymer (LiPo) batteries were selected for use to power ATLAS due to their higher energy density. They have higher discharge rates to enable more power to be drawn at once and are generally much lighter than more common battery types, such as NiCad or NiMH [29]. This makes them beneficial for remotely operated robot control.

However, LiPo batteries can pose a serious safety hazard if not handled and treated with care. When LiPo batteries are overcharged, above their rated capacity or when excessively discharged, there is a danger that the batteries will swell and become unstable [30]. This may also occur in adverse temperatures; normally above 70 degrees Celsius. If the packaging were to rupture, the exposed Lithium reacts with the moisture in the air and ignites [31]. The consequent flames and released hydrogen not only pose a risk of damaging the robot and equipment, but could also cause significant injuries to personnel within the vicinity [32]. It is for these reasons, the design of a bespoke battery monitoring system is required to avoid the LiPo batteries combusting.

5.4.1. Critical Review of Previous Designs

There has been a lack of functioning battery monitoring systems from previous WMR projects despite the safety risks posed by unmonitored and poorly-handled LiPo batteries.

The 2011/12 and 2012/13 teams did not manage to develop a functioning battery monitoring circuit. The 2013/14 team designed a circuit that employs voltage comparators to read the battery voltages. However, during simulations it was found that the circuit, triggered their LEDs, however the signal did not propagate to the desired destination. Subsequently, the team were unable to configure a design for manufacture within the time scope of the project. The 2014/15 did not consideration a method of battery monitoring nor how to shut down the power, should an issue arise, focusing primarily on methods of preventing software shutdown.

The 2015/16 WMR team designed a battery monitoring system that incorporated the use of an ATMega328 microcontroller to read the voltage of each cell of the battery [33]. The idea was to utilise a 6x10-bit ADC (analogue-to-digital) inputs to monitor the 6 cells of the battery. The ATMega328 has the capability to measure the voltage to an accuracy of 4.88mV. However, due to the cheap nature of the chip, the variation of the readings range to 3-4 times this resolution. The benefits of this include an interface for the user to monitor the voltage readings of each cell as well as having a program written to interpret the monitoring data. However, the chosen ATMega328 chip did not allow for additional battery use and would have required a new microcontroller chip for any additional batteries. This design again, did not take cell balancing in to consideration; a function that could extend battery life.

5.4.2. ATLAS – Battery Monitoring Design and Manufacture

The specifications of the batteries to be used are identified and can be seen in *Table 3*. Although not a standard SI unit, the term Capacitance (C) is commonly used in relation to batteries when referring to the capacity rating of a battery. This simply represents how fast the battery can be discharged safely and is used in combination with the known capacity of the battery [32]. For example, using ATLAS' battery information, with a capacity of 5000mAh, the capacity rating of 45C (continuously) describes how the battery can be safely discharged at 225A per hour.

Table 3: Turnigy Nano-Tech Lithium Polymer Battery Specification [34].

Specification		
Parameter	**Value**	**Unit**
Voltage	22.2	V
Capacity	5000	mAh
Configuration	6S	-
Discharge Rate (Continuous)	45	C
Discharge Rate (Maximum)	90	C
Max. Charge Rate	10	C

There are six cells within the LiPo battery being used. Each cell within a LiPo battery has a nominal voltage of 3.7V although is fully charged at 4.2V. The safety standard for LiPo batteries states that they should not be left fully charged when stored, nor should they be discharged below 3V to avoid permanent degradation of the battery [30]. Figure 33 shows the LiPo battery discharge rates for varying capacity ratings. Most manufactures set their Low Voltage Cut Off when a cell reaches 3.2V [32]. However, due to the high safety risks the LiPo batteries can pose and the close vicinity in which ATLAS will be to civilians, it was decided to keep a high safety margin and not allow the battery to discharge to below 3.3V.

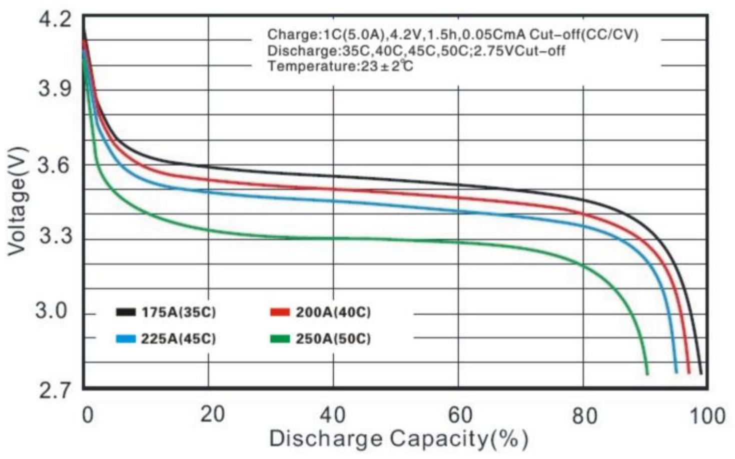


Figure 33: Lithium Polymer Battery Discharge Curve

Taking in to account WMR’s poor past experience with battery monitoring circuits, two approaches were designed; one hardware based and one software based. The first circuit comprises of a simple logic based system designed to monitor the voltage in each cell and ensure it had not discharged to below the designated safe voltage level of 3.3V. The alternative design involves a battery management chip which monitors the cell voltages via a microcontroller.

The schematic and PCB design for the hardware based circuit may be seen in Appendix A.d. The principal under which this circuit operates is that each cell is connected to a Zener diode with a reverse bias voltage of 3.3V. If the voltage within the cells drops below this, then no current can flow to a transistor. When the voltage is above 3.3V the transistor is used to send a positive voltage signal to produce a high input in the 74HC11N Triple AND gate. The AND gate acts as an intermediate between the six individual cell circuits and a non-latching battery relay. The signal sends a current through the coil in the relay, triggering the latch to close and allowing the current from the battery to flow across and carry on to provide power to the power distribution board. If a high input is received from all six cell circuits then the chip attains a logic of 1 and a signal is sent to the relay. If even one of the cell circuits drops below 3.3V then the AND gate will obtain a logic of 0 and the signal to the relay will be cut.

An L7805CV voltage regulator was used to provide the 5V power supply required for the 74HC111 AND gate and trimmer potentiometers. In the PCB design this was placed away from other components due to its tendency to heat up. After analysing component junction-to-case thermal resistance, junction-to-ambient thermal resistance and calculating the total junction-to-ambient thermal resistance, it was decided that a heat sink with a 0.83^{o}C/W power dissipation or as near to was required. This was calculated assuming V_{IN} = 22.2V and I_o = 1A.

One of the issues that comes along with this circuit is that if any of the cells even briefly fluctuate to a voltage below the threshold then the whole system will have its power cut temporarily. Additionally, the voltages of the cells can only be known as above 3.3V or below 3.3V. The actual value of the voltages are unknown and so it is impossible to know when the cells are running low but not beneath the safe voltage.

The suggestion of using an LTC6804 battery management chip for the software based circuit was proposed to counteract the first circuit's inability to provide feedback on the cell voltage values. This chip has the capability to monitor up to 12 cell voltages (two batteries) and feed this data back to a computer or microcontroller unit via a Serial Peripheral Interface (SPI) connection. The schematic and PCB design for this board can be seen in Appendix A.e

To program the LTC6802 BMS chip, it was connected to the Arduino Mega microcontroller through SPI connections [35]. To set the chip up for battery monitoring, a few initial parameters needed to be set up based on the information provided in the chip's datasheet [36].

The code is executed via four specific functions. A function that writes to the registers reading the cell voltages and sets them to zero to ensure they are empty before any data readings are taken. At the same time it also sets the Under Voltage to 3.3V by which to compare the cell voltage values to once readings are taken. The second function checks the registers were properly configured. The third function initialises voltage measurements by beginning all Analogue/Digital conversions simultaneously. In the final function, the Read Cell Voltage register group is read for 9 of the registers. Despite the voltage values being stored as 12-bits, the registers are only 8-bit. This means that each cell value is stored over 2 registers and so 9 registers need to be read instead of just 6. A calculation is done in the function to add the 8-bits from one register with 4-bits from another to find the full 12-bit cell voltage value. This is multiplied by 1.5mV to find the true voltage value as instructed in the datasheet and published on the ROS topic 'cellV'. A final function checks the flag registers to see if a flag has been raised. If any cell is below 3.3V a signal will be sent to cut off the relay to stop power to ATLAS and an error message sent.

Before each data transfer a Slave Select pin had to be set LOW to begin communication between the chip and Arduino to let the chip know the Arduino was communicating with it [37]. At the end of each data transfer section the pin was set back to HIGH to cut off communications. Additionally, each time an address command was used, the address of the chip had to be given first before stating the command byte address.

A flowchart representing this code can be seen in Figure 34 overleaf.

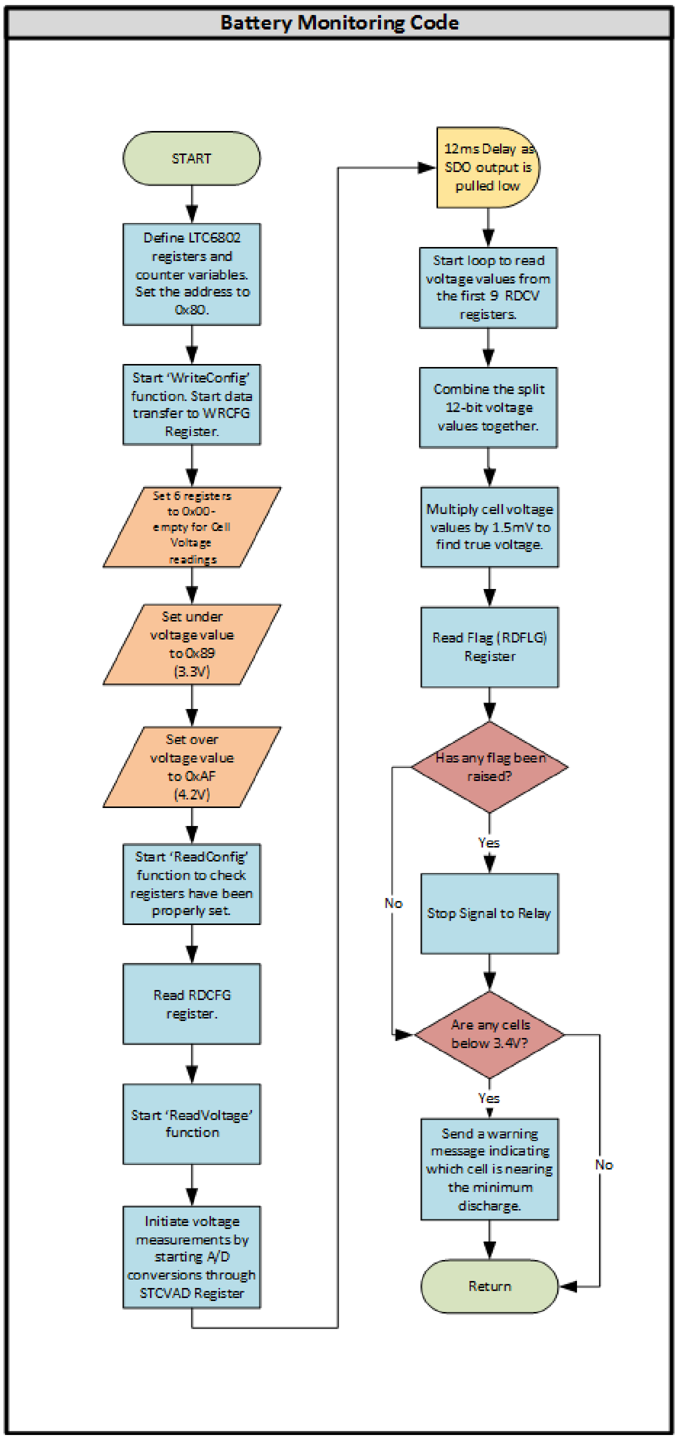

Battery Monitoring Code
START
Define LTC6802 registers and counter variables. Set the address to 0x80.
Start 'WriteConfig' function. Start data transfer to WRCFG Register.
Set 6 registers to 0x00- empty for Cell Voltage readings
Set under voltage value to 0x89 (3.3V)
Set over voltage value to 0xAF (4.2V)
Start 'ReadConfig' function to check registers have been properly set.
Read RDCFG register.
Start 'ReadVoltage' function
Initiate voltage measurements by starting A/D conversions through STCVAD Register
12ms Delay as SDO output is pulled low
Start loop to read voltage values from the first 9 RDCV registers.
Combine the split 12-bit voltage values together.
Multiply cell voltage values by 1.5mV to find true voltage.
Read Flag (RDFLG) Register
Has any flag been raised?
Yes
No
Stop Signal to Relay
Are any cells below 3.4V?
Yes
No
Send a warning message indicating which cell is nearing the minimum discharge.
Return

Figure 34: Battery Monitoring Flowchart

5.4.3. Innovation – Optimised Battery Monitoring and Added Battery Balancing Capability

The proposed LTC6802 chip incorporated in to this year's design can read the cell voltages to an accuracy of 1.5mV/bit. Additionally, due to its stackable architecture, multiple cells can be connected in series and daisy chained without additional optocouplers or isolators [35]. Thus, removing the prerequisite of the 2015/16 board for a new microcontroller chip should any additional battery be included. The LTC6802 is able to sample all the cells in just 13ms; a process that took the ATMega328 300ms - over 20 times as long. This means more readings can be taken and an even more accurate picture of the battery's voltage levels can be created.

In the future, the plan is to have two batteries connected in parallel. The topology provides a more stable power supply and prevents too much voltage drop. Each battery will be attached to its own relay so that the robot is still able to continue running should any of the batteries fail.

The additional capability of the LTC6802 chip should be utilised to give the added potential to balance the batteries as well as just monitor them. This will be beneficial as it will allow the cells to discharge at the same rate and avoid overcharging any of them which can occur if any cell discharges at a much faster rate than the rest.

An additional emergency stop is to be incorporated as a hardware cut off should any of the team personnel feel the need to cut the power being supplied to the system.

5.5. Overcurrent Protection Circuit

It is the purpose of an overcurrent protection circuit or protective relay to prevent specific elements of the system from damage. Typically, protection schemes utilise Kirchhoff's Current Law that states, *"the algebraic sum of all the currents flowing into any junction in a circuit is zero"* [38]. Observing the current flowing within a system offers insight into how the system operates. Protection circuits may operate by identifying current deviations in circuit *'nodes'*. When current levels exceed the expected level, deviating from nodal stability, it is indicative of abnormal current paths. Excesses current suggests that one or more elements within the system are consuming more power than expected [39]. Similarly, if the current is lower than expected, it may suggest that an element within the system is not powered correctly or may be disconnected.

ATLAS requires overcurrent protection circuitry as the motor controllers, RoboClaw 2x15A, selected do not consist of the necessary protection. The variety of terrain and obstacles that ATLAS will face will result in uneven and irregular current draw, which must be monitored in order to ensure safe and efficient operation.

5.5.1. Critical Analysis of Previous Designs

The 2015/16 project, Cyclone, used Maxon ESCON 50/50 P/N 409510 motor controllers [40]. These motor controllers also require external overcurrent protection, for the reasons stated above. *Maxon* suggests, *"that the overcurrent protection circuit is configured inoperative within the* *operating range"* [41]. The continuous output current is recommended to be 5A, whilst maximum current draw is noted to be no more than 15A for a time period no longer than 20s [41]. It must be noted that the Cyclone team's overcurrent protection consisted of a 20A Littelfuse [40]. It is therefore apparent that that the motor controllers would reach their maximum operating limit and become destroyed, before the fuse would blow, cutting power to the robot. Fuses are commonly used as a means for short-circuit protection. However, they must be replaced in order for the system to operate again. Furthermore, a significant factor that limits the effectiveness of fuses is the time-current dependency [39]. This is beyond the scope of this report, however, more information may bay be found within literature [39] [42].

5.5.2. ATLAS' Overcurrent Protection Circuit – Design and Manufacture

An alternative method, becoming ever more common, is to allow the system to protect itself. If the current is detected outside the normal operating conditions, not only does the system protect itself, but is able to return to 'normal' after the *"the fault condition has been cleared"* [39]. This method of overcurrent protection utilises a comparator, which monitors the operating current to a defined threshold. Such a method would allow ATLAS to be extensively tested, without the need for large volumes of fuses and the worry of damaging the system.

Several overcurrent protection topologies were considered and simulated. Products from Texas Instruments [43] include INA199 and INA300. After iterative design and empirical simulation, it was decided to use an INA300, *"a specialized current sensing comparator with the ability to perform the basic comparison to expected operating thresholds required for out-or-range detection"* [39]. This device's use of integrated reference points within the comparator that dictate the threshold trip point via a single external resistor, make it the preferred option. A latch configuration is also available. This is beneficial when the state of the output cannot be continuously monitored. The basic operation of this device may be seen below in Figure 35.

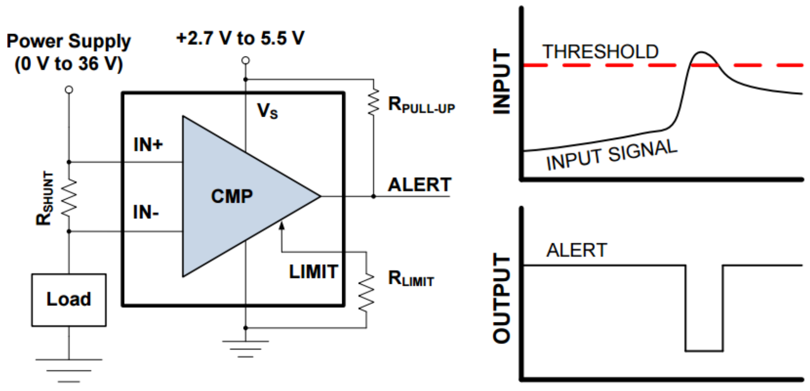


Figure 35: INA300 Overcurrent Comparator [39] .

The INA300 measures the differential voltage across IN+ and IN-. The shunt resistor is dependent upon the current requirements within the system. The differential voltage is then compared to the threshold voltage, which is dictated by the user. A 1k POT was inserted to give system flexibility and aid testing. The R_{LIMIT} was set to dictate a threshold voltage of 30V. Preliminary experimentation on the motors resulted in 110W power drawn from the system. It must be noted that one of the motor shafts was bent. Consequently, the circuit had to be adapted to accommodate the additional current drawn from this motor. As may be seen in Figure 35, the alert is pulled low when the threshold current is exceeded. The INA300's response results in a current excursion within 10µs. Typically; the Alert, Latch and Limit may be controlled via microcontroller. However, it was deemed sufficient at this stage within the project to use a 20A relay as a replacement for the fuse used in the Cyclone system. As may be seen within the circuit, several switches have been included with the circuitry. These simply offer the WMR team the ability to bypass elements of the design to offer increased flexibility and ease of testing. It became apparent from analysing the previous WMR project that component tolerance and safety margins were often neglected, resulting in electronics not being able to operate. To overcome this, it was agreed with the aid of specialists that an additional 20% should be added to all calculated components; offering a safety buffer. The overcurrent protection circuit protects the system from current beyond the operational range. The schematic and PCB design for this can be seen in Appendix H. Further testing and analysis performed led to the conclusion that the INA300 should ideally be controlled via a microcontroller. It is important to note that it is not an overload protection circuit which prevents excessive heat and burning out components. An overload protection circuit is not required as the current should be cut before this occurs.

5.5.3. Innovation – Self Monitoring and Programmable Future Capabilities

Further flexibility to the system is offered by implementing the configurable INA300 chip. This will allow for future WMR teams to gain feedback on the current draw of the motors as well as protecting against overcurrent. It is recommended to future WMR teams that component footprints and standards made observed prior to purchase. It is good practice to have all your components of a similar size. The INA300 is 2mm x 2mm and much smaller than any other device. This became challenging when designing the PCB layout.

5.6. Communications

For the teleoperation of ATLAS, it is necessary to consider the method of communication. The use of a Wi-Fi connection is one of the most popular methods for data transmission over a local area. The 802.11n Wi-Fi protocol is widely used because of the dual-band ability of transmitting signals at frequency 2.4GHz and 5GHz [44]. Older Wi-Fi enabled devices typically use 2.4GHz and interference on this frequency from electrical devices such as microwaves and garage door openers are also problematic. The channels used on this frequency also interfere with each other. In contrast the more modern standard has channels on the 5GHz frequency spaced well apart such that there is no interference between channels. The channels have a greater bandwidth available to them and when combined with less interference leads to a greater available data transmission rate.

The main drawback, however, of using 5GHz channels is that the higher frequency signal struggles to penetrate high density obstacles such as concrete walls or floors, resulting in heavily-weakened signal strength and thus a lesser range. It is for this reason the 2.4GHz frequency has been selected as a more robust network connection is favourable over greater transmission speed.

5.6.1. Critical Review of Cyclone's Router:

The Cyclone project chose the compact Buffalo AirStation™ AC433 wireless travel router as the device to handle network communications. Its lightweight and small form factor were desirable traits. The voltage requirement of 5V made it a suitable solution for wireless communication as it can be connected directly to the USB ports of the on-board computer without a separate power supply. This reduces the complexity of designing the power supply and installation. It is for these reasons it has been selected and to be reused as the on-board Wi-Fi router of ATLAS. Antennas have also been attached to the router to increase wireless performance.

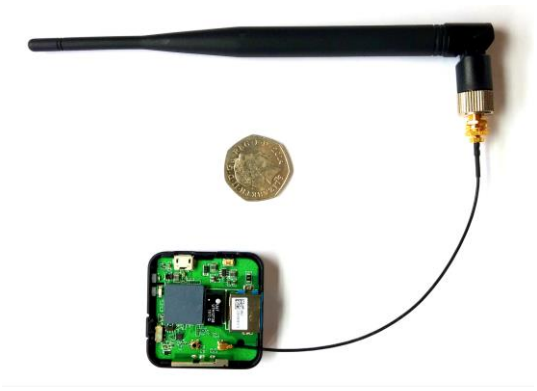


Figure 36: Size of the Buffalo router relative to a 50p coin, showing the retrofitted antenna

5.6.2. ATLAS - Data Rate Considerations

The data rate of live streaming the uncompressed video captured by the front and rear web cameras can be calculated with the The Kush Gauge™ formula [45]:

$$\text{Number of devices} \times \text{image width} \times \text{image height} \times \text{FPS} \times \text{motion rank} \times \text{constant} = \text{final bitrate in bps} \quad [5]$$

Using the PlayStation Eye webcams at 640×480 pixels and 24 Hz:

$$2 \times 640 \times 480 \times 24 \times 2 \times 0.07 = 2.064\ Mbps \quad [6]$$

To allow for smooth live streaming when a higher frame rate is required, it is recommended to reserve at least 3Mbps.

It was experimentally observed that this bandwidth, summed with the total bandwidth required from other sensors (such as LiDAR, IMU, control signal etc.), does not exceed 5Mbps. Subsequently, to accommodate the future addition of a robot arm, higher resolution web cameras and 3D mapping tools, the data rate bench mark has been set to 10Mbps.

5.7. ROS programming

Robotic Operating Systems (ROS) was selected as the primary development suite within which to interface the electronic hardware and sensors. ROS is an open-source, easily accessible operating system used to support the rapid development of robot software [46]. ROS runs on Ubuntu supported platforms. It is independent of any programming language although the main libraries used in conjunction with it are Python and C++. It provides low-level device control, passes messages between processes and it functions using a peer-to-peer network of processes known as nodes. [47]

Nodes are set up with each being a piece of executable code responsible for a specific task. These can be written in any language. The benefit of this is that professionals in unique fields can publish complex nodes that can be accessed by the ROS community and integrated in to their own systems. The nodes are able to communicate with each other by publishing and/or subscribing to data packets known as topics. The data types of each topic are independent of each other so any data format can be communicated. This data is also exchanged asynchronously so that multiple nodes can communicate information on the same topic. A network of nodes are all brought together via a Master. The master acts like a central database to store parameters and information from different nodes to be shared with each other. The ROS master utilises an XMLRPC form parameter service. XMLRPC is a stateless, HTTP-based protocol which results in each packet of data being considered to be unrelated. This means the ROS architecture allows for distributed computing and so is able run nodes on separate hardware and form a robot system. Since ATLAS's system will be comprised of multiple computers communicating through a wireless router, ROS's distributed computing ability makes it an ideal operating system enabling communication between the on-board computer and the master laptop to be significantly easier.

Data from these nodes can be stored as bags to be reviewed as many times as one needs at a later date. This is very useful for testing and debugging and is a key feature that was considered for going to RoboCup, as the competition requires a submission of data files to score points.

ROS provides a simple platform on which the various sensor readings, battery monitoring readings and control commands for ATLAS can be easily accessed, as well as reviewed post readings.

5.8. Control

With the decision that ATLAS's newly improved design would include flipper arms on the front and back, basic robot control was required to include both inner track driving and flipper arm rotation. This differed from the 2015/15 WMR team's design whose robot only required two track control.

5.8.1. Hardware

The hardware making up the motor control is broken down in to the joystick controller and motor controllers.

Controller Choice

It was decided that a Sony PlayStation 3 (PS3) controller would be used to drive ATLAS. The PS3 controller was chosen for its ergonomic and fairly robust design. The two analogue sticks on the PS3 controller drive the two inner tracks with the left and right joysticks controlling the left and right tracks respectively. The left buttons on the top of the controller were chosen to control the front flippers and the right top buttons were chosen to control the rear flippers. L1 and R1 rotate the flippers in the forward direction and L2 and R2 rotating the flippers in the reverse direction. This can be seen in Figure 37. It was decided that the flippers would be set to rotate at a pre-set speed instead of varying at a speed dependent on the amount of button depression, as this would give the user greater control of the flipper rotation and allow for better tracking of the flipper's orientation when not in the user's line of site.

The PS3 joysticks could be easily mapped through ROS and the top L1, L2, R1 and R4 buttons made for a suitable choice for accessible flipper control to rotate each set of front and rear flippers.

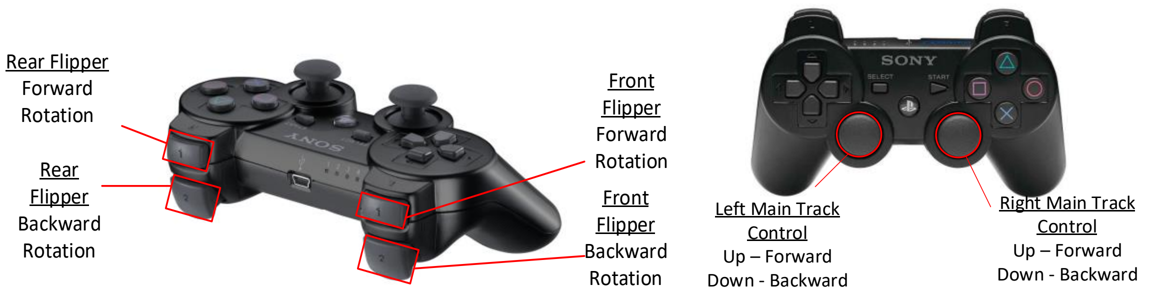

Rear Flipper Forward Rotation
Rear Flipper Backward Rotation
Front Flipper Forward Rotation
Front Flipper Backward Rotation
Left Main Track Control Up – Forward Down - Backward
Right Main Track Control Up – Forward Down - Backward

Figure 37: Playstation3 Controller Button Controls

Table *4* shows how the buttons map through ROS and which pin the signal is being sent out of through the Arduino Mega.

Table 4: PS3 Controller Button Mapping

Button	ROS Mapping	Arduino Output Pin	Comments
Left Joystick	1	10	
Right Joystick	3	12	
L1	8	14	Same output as L2 but signal is in reverse direction
L2	10	14	See L1
R1	9	16	Same output as R2 but signal is in reverse direction
R2	11	16	See R1

Motor Controller Selection

Due to the addition of the flipper arms and hence two additional motors, it was determined that new motor controllers had to be ordered to accommodate the control of the flippers. The existing controllers used by the WMR 15/16 team were originally purchased by the WMR 14/15 team, chosen for their compatibility with both the motors and the chosen microcontroller (Arduino Mega), their efficiency in limiting and managing the voltage and current input of the motors and their precision [48]. These criteria were used as a basis for motor controller selection for the flippers. Moreover, the motor controllers had to be able to handle the voltage input from the battery. Several motor controllers were compared against the requirements and the current controllers as seen in Table 5.

Table 5: Motor Controller Requirements and Comparison

Option	General Information				Electrical Characteristics		I/O Capability	
	Motor Controller	**Dimensions (mm)**	**Weight (g)**	**Cost (£)**	**DC Voltage (V)**	**Continuous Current (A)**	**USB Configurable**	**Arduino Compatible**
1	Maxon ESCON 50/5	115 x 75.5 x 24	204g	£205.40	10 - 50	5	Yes	Yes
2	RoboClaw 2x15A	74 x 52 x 17	61g	£72.79	6 - 34	15	Yes	Yes
3	Sabertooth Dual Regenerative Motor Driver	60 x 80 x 20	96g	£124.99	6 - 30	25	No	Yes

The team felt when given the choice, dual motor controllers were a better alternative to single motor controllers. Dual controllers would reduce the number of motor controllers required by half whilst also minimising the connections that needed to be made for communication. With their single motor capability, this was the area where the Maxon ESCON controllers fell short [49]. To use the ESCON controllers meant that not only would two controllers be needed for each of the inner tracks but an additional two to drive each of the flipper motors. This would add an extra 400g to the robot as well as being extremely over budget. The Sabertooth motor drivers were not only able to deal with the required current and voltage but also offered the addition of being able to offer regenerative abilities to charge the motor when the motors were reversing or moving at slow speeds [50]. Sabertooth was a lighter controller option at a lower cost than the ESCON controllers but unfortunately did not have USB interface. The RoboClaw motor controllers have dual motor capability, could handle both the current and voltage requirements and had a USB interface to allow configuring, diagnostics and monitoring of the connected system through a computer [51]. Weighing in much lighter than the other two controllers and at a lower cost whilst still meeting the necessary requirements, it was decided the RoboClaw controller would be purchased.

Following this decision, it was found that the already owned ESCON controllers weren't functioning properly and so the decision was made to purchase two RoboClaw controllers. One to control the inner tracks by replacing the two ESCON controllers and one to control both flipper arms. This would also keep with continuity within the robot by keeping the type of motor drive processor consistent.

RoboClaw Configuration

The RoboClaw 2x15A Motor Controllers are connected to receive voltage inputs from the battery of 22.2V and inputs from the Arduino Mega via its supported 3.3V/5V logic pins. Using the downloadable IonMotion software to support the RoboClaw, the controller was configured to be in Packet Serial Mode. This mode enables the controller to receive more comprehensive serial input signals from the Arduino in Pulse-Width-Modulated (PWM) form. These PWM signals are used to set the velocity of the individual motors. The benefit of PWM signals, like with most digital signals, is it is insensitive to noise which when sending control signals is important to try and avoid.

5.8.2. Software

The main code to drive ATLAS was written across using a combination of ROS, C++ and the Arduino interface. The joystick node connected to the laptop was read using ROS software, the relevant values selected and then published on the topic 'twist' across the Wi-Fi connection to the Pico computer to then be processed by the Arduino (Figure 38).

Node Control and Interaction

Three nodes were created to run ATLAS' motor controls; a 'Joy' node, a 'Drive' node and the ROS_Serial node. This is represented in the diagram shown in Figure 38, which also shows the topics 'LeftTrack' and 'RightTrack' being monitored.

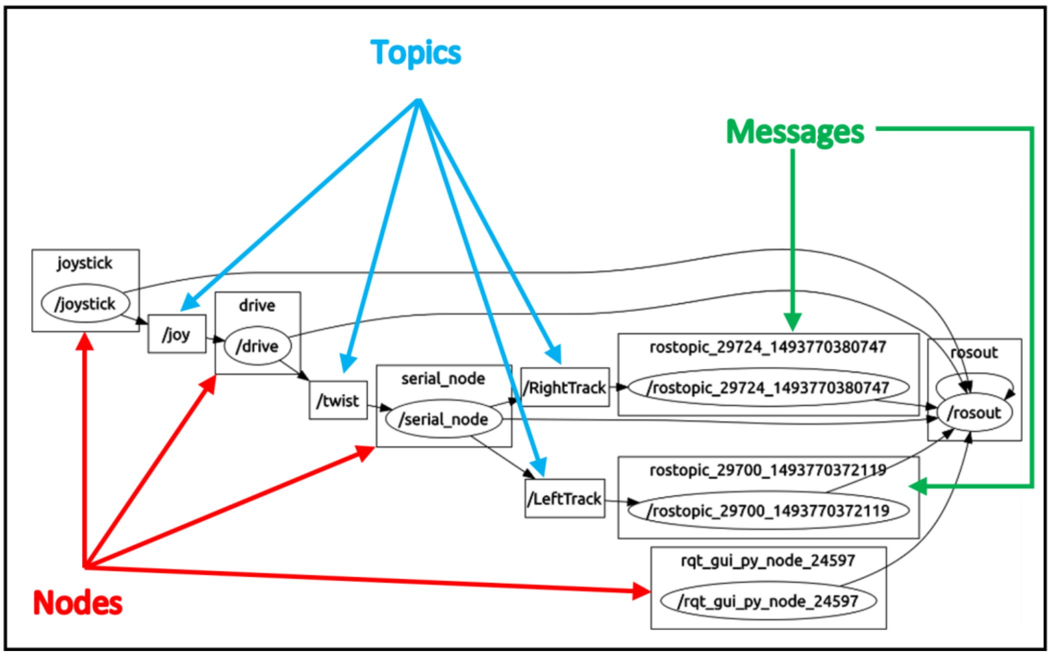


Figure 38: rqt_graph of ATLAS Node Connections for Left and Right tracks

The ROS_Serial node is a standard node used amongst those using ROS to interact with a chosen microcontroller. For ATLAS, this node was setup to handle communications with the Arduino Mega. It provides a ROS communication protocol that works over the Arduino's UART and allows the Arduino itself to function as a node capable of publishing and subscribing. This sets up the Arduino to be able to then subscribe to the topic ‘twist’ as well as publish the sensor data and battery monitoring data over the ‘CO2Values’ topic and ‘Voltage’ topic respectively.

The PS3 controller was connected to the master laptop and interfaced with ROS through the ‘Joy’ node. This node decodes the incoming data received from the controller and publishes it on the topic ‘joy’. The data published contains the button information in both axis and button form with each button having been assigned an appropriate ID number. The diagram for this can be seen in Appendix I.

The drive node was written in C++ and subscribes to the ‘joy’ topic. The controller data was sent from the joy node as a 16-bit integer which is then converted and scaled. For the inner tracks, the data is converted in to a scaled floating-point number between -1 and 1. For the flipper arms, the button data is converted to a scaled integer value between 0 and 1. This data is then published on a topic named ‘twist’.

Main Tracks

This scales the received floating point values from a -1 to 1 to a 0 to 255 scale to match that of a PWM signal range stored in variables m1_raw and m2_raw. Equation 7 shows how the values were scaled.

$$mx\_raw = (motor \times -127) \times 127) \qquad [7]$$

These values are run through a limiting function which was included to prevent the motors from overrunning and drawing too much current. This means that the motors should only operate between a 90% and 10% threshold and thus prevent stalling. The RoboClaw motor controllers make use of their own Arduino library functions to send signals to indicate whether the robot should go forwards or backwards. The value of each function should be between 0 and 127, with 0 being STOP and 127 equalling FULL FORWARD or FULL REVERSE depending on which function the signal is being sent through. For this reason, the new values are then checked to see if they’re above or below 127. Values above 127 will indicate that a reverse signal has been sent through the joystick. The reverse signal values will be in the range of 127 to 255, with 127 being STOP and 255 being FULL REVERSE. Values below 127 will indicate a forward signal has been sent through the joystick. These are in the range of 0 to 127, however, due to how the data was scaled, a value of 127 would designate STOP and 0 would mean FULL FORWARD. Equation 8 and Equation 9 show how the final command values sent to the RoboClaw were calculated to accommodate the RoboClaw function requirements.

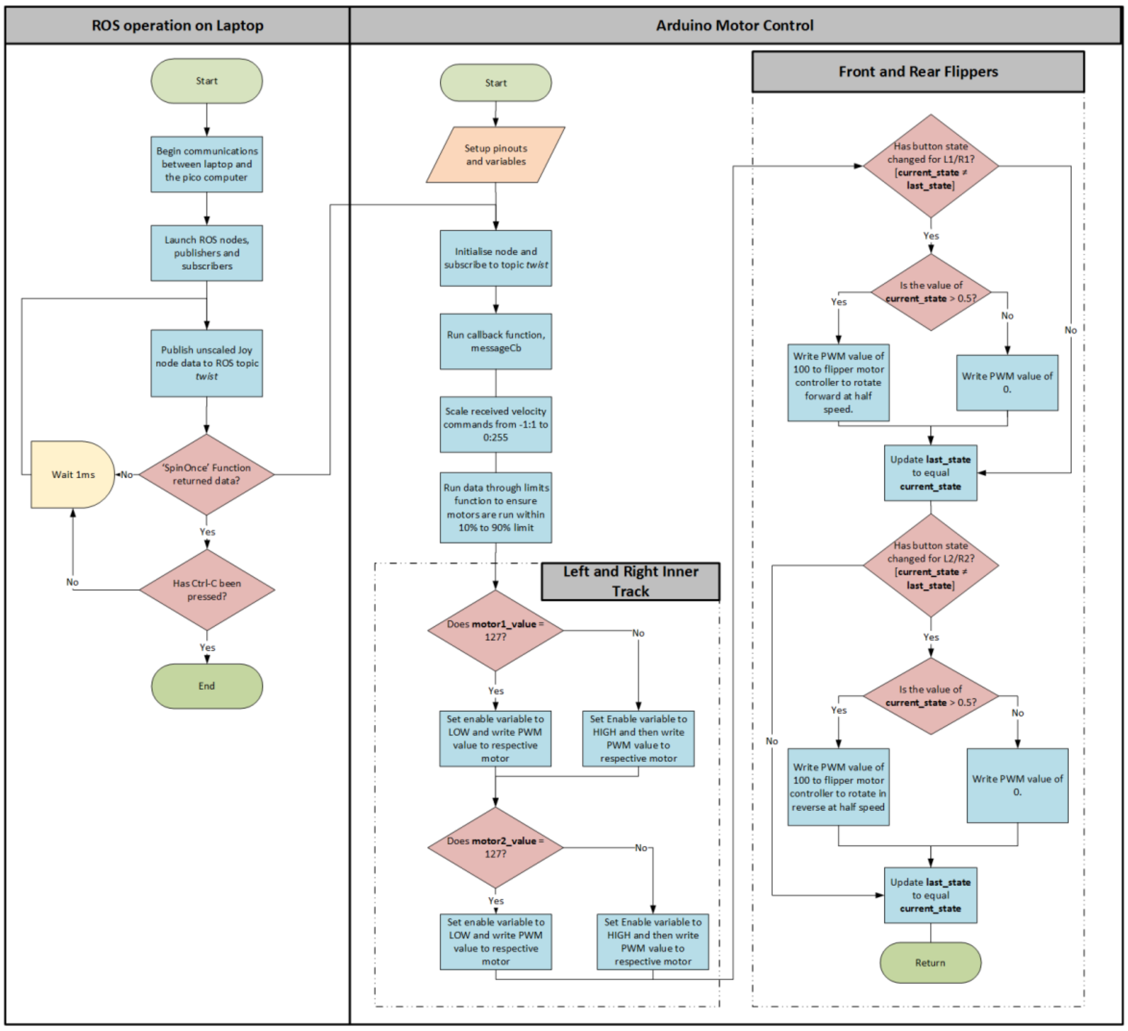
$$Forward\ velocity\ control\ value = 127 - motor\_value \qquad [8]$$

$$Reverse\ velocity\ control\ value = motor\_value\ -\ 127 \qquad [9]$$

Flippers

For the flippers, a debounce function was included that checks the state of the button in question, compares it with its previous state to check whether it was just a fluctuation or not and then performs the required act depending on if the change of state is constant. If it is confirmed that the button has been compressed then a set PWM signal of half speed will be sent to the motor controllers; in this case a standard speed of 64 would be written to the either reverse or forward function. It was decided that the flipper's velocity did not require variation and could be allowed to go at a constant velocity in either the forward or reverse direction.

Figure 39 gives a visual representation of how the motor control works.



ROS operation on Laptop
Arduino Motor Control
Start
Begin communications between laptop and the pico computer
Launch ROS nodes, publishers and subscribers
Publish unscaled Joy node data to ROS topic twist
'SpinOnce' Function returned data?
No
Wait 1ms
Yes
Has Ctrl-C been pressed?
No
Yes
End
Start
Setup pinouts and variables
Initialise node and subscribe to topic twist
Run callback function, messageCb
Scale received velocity commands from -1:1 to 0:255
Run data through limits function to ensure motors are run within 10% to 90% limit
Left and Right Inner Track
Does motor1_value = 127?
Yes
No
Set enable variable to LOW and write PWM value to respective motor
Set Enable variable to HIGH and then write PWM value to respective motor
Does motor2_value = 127?
Yes
No
Set enable variable to LOW and write PWM value to respective motor
Set Enable variable to HIGH and then write PWM value to respective motor
Front and Rear Flippers
Has button state changed for L1/R1? [current_state ≠ last_state]
Yes
No
Is the value of current_state > 0.5?
Yes
No
Write PWM value of 100 to flipper motor controller to rotate forward at half speed.
Write PWM value of 0.
Update last_state to equal current_state
Has button state changed for L2/R2? [current_state ≠ last_state]
Yes
No
Is the value of current_state > 0.5?
Yes
No
Write PWM value of 100 to flipper motor controller to rotate in reverse at half speed
Write PWM value of 0.
Update last_state to equal current_state
Return

Figure 39: Motor Control Flow Chart

Extra code was written in to smooth the signal being sent to ATLAS to avoid sudden high accelerations. An example of this is seen in Figure 40.

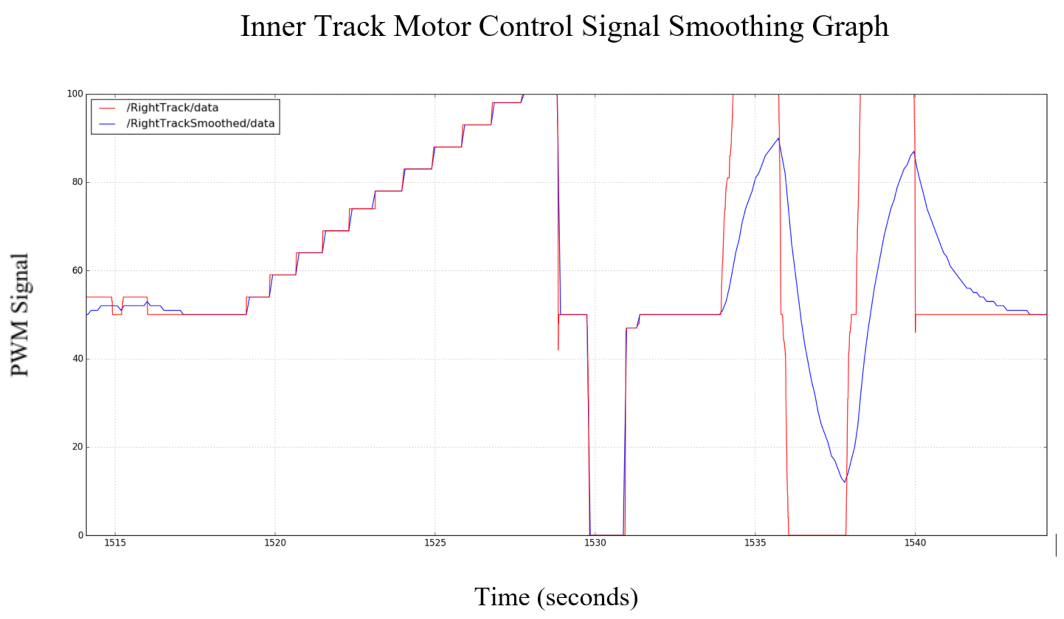


Figure 40: Graph of Motor Control Signal Smoothing

Once the initial motor and flipper control was achieved, additional functionalities were then added to the controller. The up/down arrows were used as throttle control. The increase of this can also be seen in the first 1500 seconds of the graph in Figure 40. Torch control was incorporated, so that a light on the front of ATLAS would be turned on if selected. This toggles through low power, medium power, high power and strobe lighting. See Appendix E for more details. Finally, an ON/OFF button was included to ensure the motors would not accidentally be driven when not in use.

5.9. Sensors

Sensors are a way for robots, like ATLAS, to gain an understanding of the environment they're in. They provide feedback with regards to the robot and/or its surroundings for the robot or the person operating the robot to interpret. To be able to compete in the RoboCup, certain sensing capabilities are required as will be explained in Section 6.2. ATLAS' sensor array, as briefly shown in Figure 32, comprises an Xbox Kinect, a Light Detection and Ranging (LiDAR) sensor, front and rear cameras, an Inertial Measurement Unit (IMU) and a Carbon Dioxide (CO_2) sensor. Further details on the individual sensors can be found in Appendix D.

5.10. ATLAS App

With the idea of this year's WMR project built around providing a stable platform from which future years can develop the robot rather than starting from the beginning each year, it was decided an intuitive graphical user interface which required limited training and development was necessary.

Using the free integrated development environment Corona SDK [52], a complete user interface has been built in the programming language of Lua in around 1,000 lines of code, something unobtainable within the time frame using the ROS package "rqt" [53] and python/C++. Corona SDK allows the same code to be compiled for Windows, Android, iPhone, iPad, Mac OS and other mobile device and computer platforms. Lua's easiness to learn, coupled with the extensive application programming interface (API) documentation means future years will be able to adapt the software written this year without a strong background in programming.

The main menu, shown in Figure 41, shows all current features: main track drive, flipper arm actuation, IMU feedback, telemetry display, dextrous arm Inverse Kinematics, battery monitoring, camera streaming and CO2 sensor monitoring.

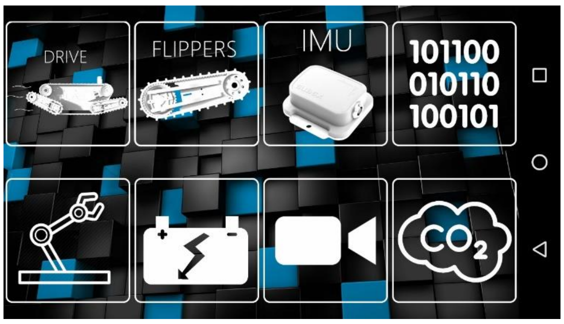


Figure 41: The ATLAS App - Main Menu

The application can serve as a back-up to the more practical gamepad controller in actuating all four motors. The app connects through WiFi to the main laptop which runs the C++ wrapper, Socket [54], within a ROS node which listens to the app. This node then publishes the data to ROS. This allows the user to send data to ROS; in this case the tracks and flipper motor control signals. Held in landscape mode, the left thumb is used to drive the robot's main tracks forward and backwards. The right thumb can then rotate the image of the robot clockwise and anticlockwise to control the track differential and cause it to rotate. The numbers at the sides of the two robots range from -100 to 100 and represent the percentage of maximum power the motors run at. As seen in the left in image in Figure 42, the robot is rotated anticlockwise at a value of -26, meaning the left track will reverse at 26% and the right track will drive forward at

26%. Simultaneously driving the track either forwards or backwards with the left thumb would then cause the robot to move in a circle rather than rotating on the spot.

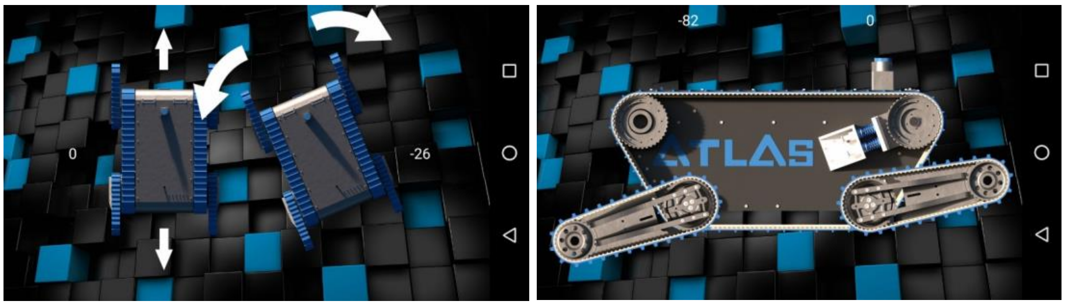
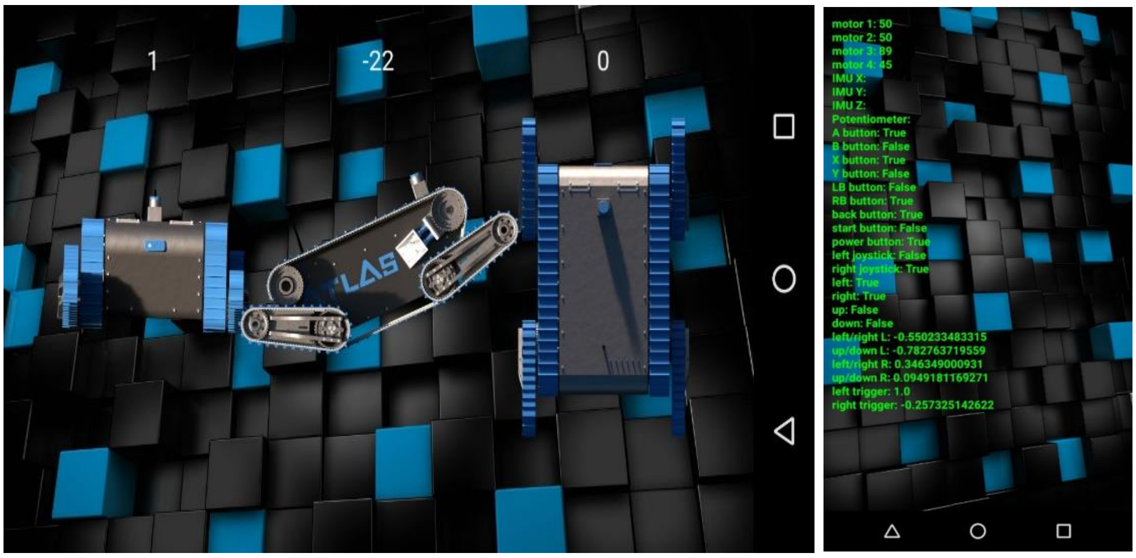


Figure 42: The ATLAS App – Track Control (Left) and Flipper Control (Right)

The flipper arms can be driven in a similar fashion by rotating the images of the flipper arms in the app, as seen in the right side of the figure. The rear track here is rotated anticlockwise to a value of 82%. Future years, with the use of encoders, can use this app to drive the flipper arms to an angle.

It is anticipated this app will be used in a distributed manner with multiple users connected to the main computer at once. This allows users to each monitor one subsection of the robot. A visual representation of the robot's orientation is shown in Figure 43. This screen will be particularly useful to the robot's operator who can visualise the robot's roll, pitch and yaw and adapt the handling of the robot as required.



```
motor 1: 50
motor 2: 50
motor 3: 89
motor 4: 45
IMU X:
IMU Y:
IMU Z:
Potentiometer:
A button: True
B button: False
X button: True
Y button: False
LB button: False
RB button: True
back button: True
start button: False
power button: True
left joystick: False
right joystick: True
left: True
right: True
up: False
down: False
left/right L: -0.550233483315
up/down L: -0.782763719559
left/right R: 0.346349000931
up/down R: 0.0949181169271
left trigger: 1.0
right trigger: -0.257325142622
```

Figure 43: The ATLAS App – Visual Representation of the robot's orientation (Left) and Telemetry Pane (Right)

Here the app is connecting to another ROS node which is running a python package named SimpleHTTPServer [55]. The node listens to all necessary topics and saves the list of messages from those topics to a local text file. The server then listens to requests coming from the app and sends the text file at a rate of 50Hz. To show all the data in the most basic form, a telemetry pane is provided, also shown in Figure 43. Currently the majority of the data transmitted is the instantaneous values of the variable controller inputs, such as the analogue joysticks, and the buttons, which have been set up to serve as Boolean toggle switches.

Previous years overlooked the software control of the articulated arm. Despite its use being outside of the scope of this year's project, the arm's inverse kinematics have been calculated and applied in the app. This can be seen in the left image of Figure 44. The robot's head can be dragged by the user to the desired position. Future developments of this app will require the azimuthal rotation of the arm base, the tilt of the arm's head and the actuation of the end-effector to all be controlled from the app.

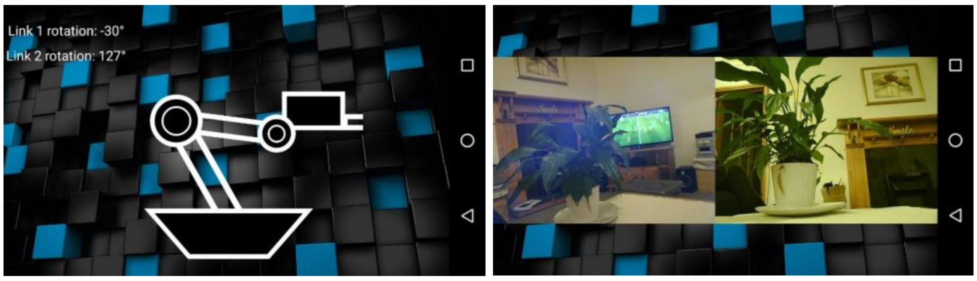


Figure 44: The ATLAS App – Robot Arm Inverse Kinematics (Left) and Video Feed (Right)

Using the ROS package usb_cam [56] as the hardware driver, previous years' webcams can be connected to ROS to publish image data. Another package, web_video_server [57], is then used as an HTTP server which transmits these images in MJPEG (motion jpeg) or webm video format upon request from the local network. As shown in Figure 44, the app is then set up to display two video feeds at 24 frames per second side by side, or focussing on one camera by tapping on it.

An example of graphing is shown in Figure 45. Here the app is listening the data being sent through SimpleHTTPServer from a node subscribed to the CO2 sensor. The graph's code is easily reconfigurable, and its X and Y ranges are automatically updated every frame to ensure the full data fits onto screen in the best manner.

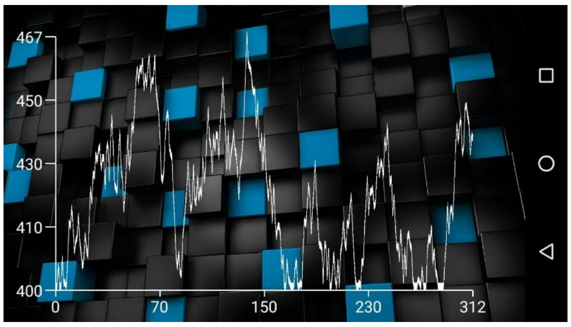


Figure 45: The ATLAS App – Graphing Example

The flexibility of Corona SDK, and the ease of programming in Lua, means future years should be able to extend the capabilities of this app to be a more advanced control system for Atlas. Specifically, displaying mapping data from the LiDAR sensor and drawing a path for the robot to follow would be a significant achievement.

5.11. Electrical and Software Summary

To improve on the power board design from last year, a modular approach was taken towards the power electronics and four separate proposed boards designed to be able to work together whilst still being standalone component. These covered the areas of power distribution, battery monitoring and overcurrent protection. Unfortunately, delays in manufacturing meant that by the time of report writing, only three of the four boards had been machined and were yet to be populated. These can be seen in Figure 46.

With regards to communication, the decision was made to continue using the Buffalo router selected by the WMR 15/16 team. Additional bandwidth was provided to accommodate future components.

Control of ATLAS was primarily executed using ROS. Motor control for the tracks and flippers were achieved with extended control functions included. An array of sensors were implemented, the testing results of which can be in Section 6.2 Exploration Testing.

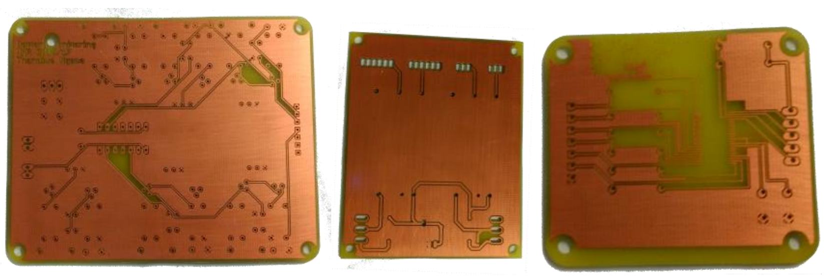
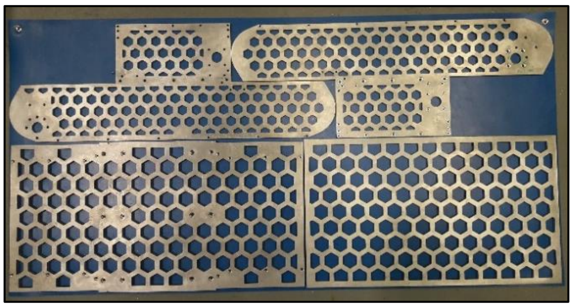
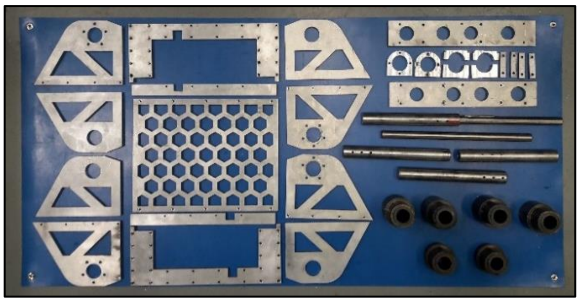


Figure 46: Un-Populated Circuit Boards

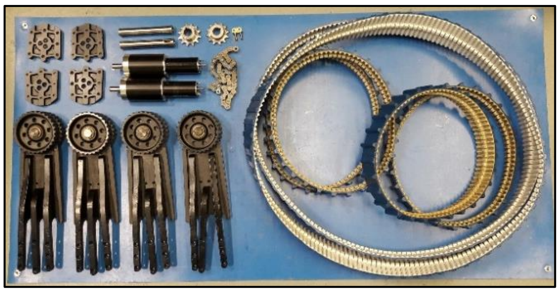
Final Design – ATLAS



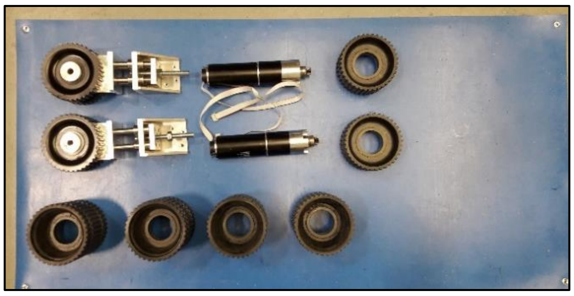
(a)



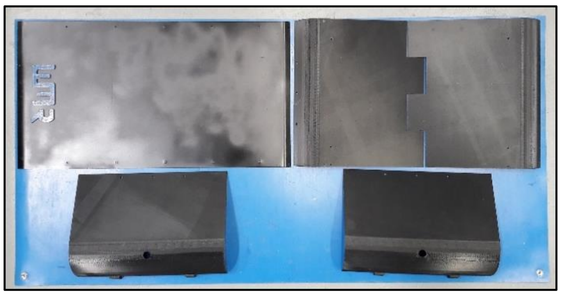
(b)



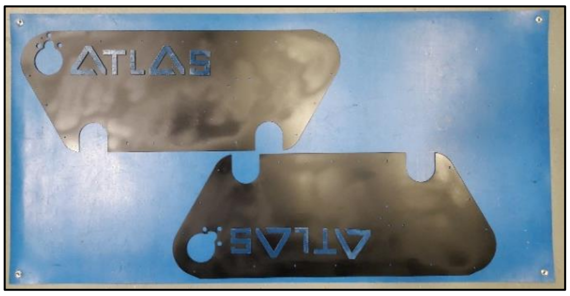
(c)



(d)



(e)



(f)

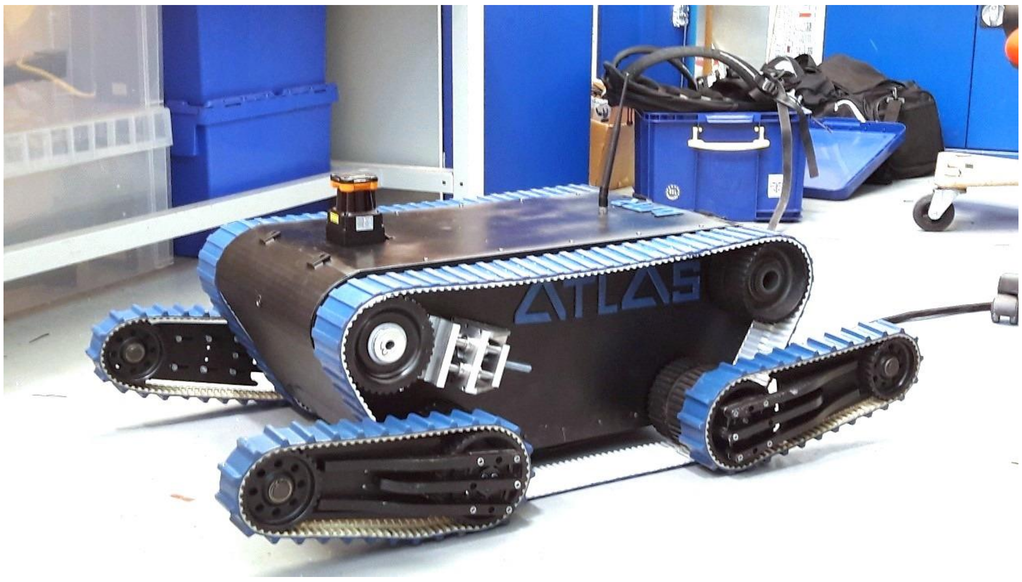


Figure 47: ATLAS Final Assembly (a) Main Chassis, (b) Lower Chassis & Suspension, (c) Flippers and Tracks, (d) Tensioning and Drivetrain, (e) Cladding Plates (f) Side Cladding & Logo

6. Real-World Testing

After ATLAS was successfully manufactured, it was time to test its capabilities in real-world environments to gauge whether it would be capable to enter to RoboCup competition. All tests conducted were in compliance with the types of challenges ATLAS would face at RoboCup as detailed in Section 8.3.

6.1. Manoeuvrability and Mobility Testing

Manoeuvrability and mobility tests consisted of the most basic functions of ATLAS. This was the ability to move forward, backwards, turn on the spot and raise itself up on its flippers. All control was done using the PlayStation 3 controller as intended however an umbilical cord was required for power. Initially, the intention was for the flippers to rotate at a half speed. However, as soon as testing began, it was quickly found that the flippers required rotating at, at least 78% of their maximum speed to generate enough torque to lift ATLAS. Once this was rectified, ATLAS was easily capable of performing all of the tasks as intended. Concern was, however, raised over the maximum speed which reached only 0.2m/s on flat ground. Figure 48 shows time-lapses of ATLAS performing these operations.

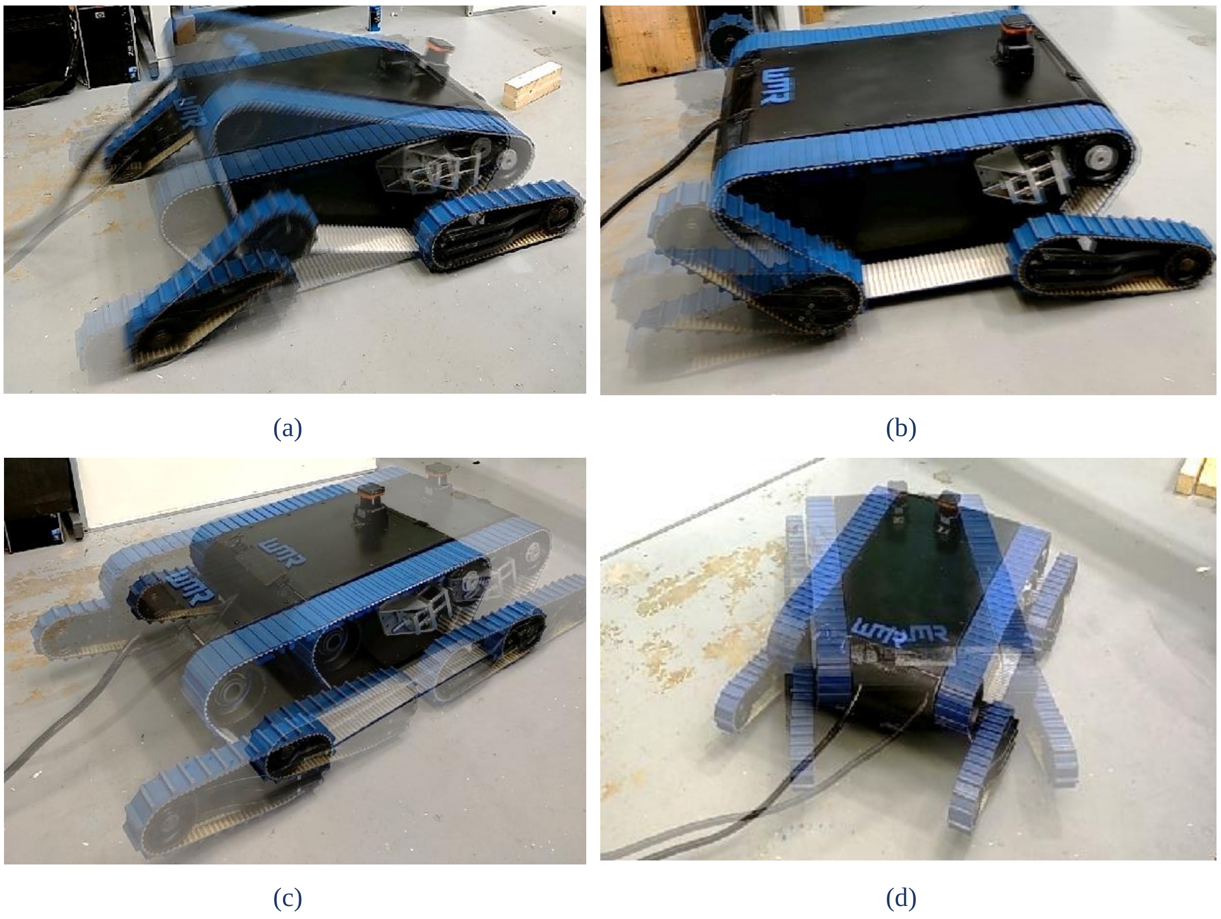


Figure 48: ATLAS Successful Testing Operations; (a) Lifting (b) Flippers (c) Forward Movement (d) Turning

More advanced tests involved navigating around obstacles, which were unfortunately not available. The key test of ATLAS' ability was climbing stairs as can be seen in Figure 49.

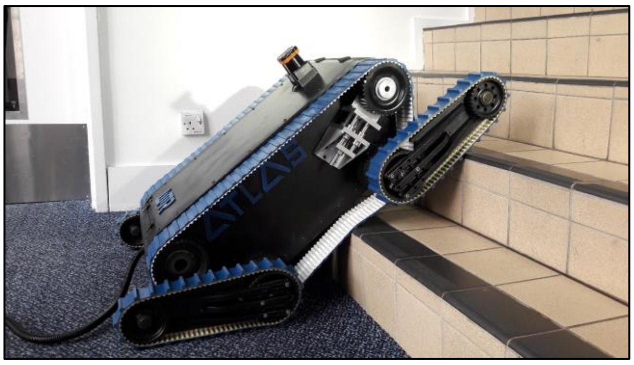


Figure 49: ATLAS Climbing a Staircase

ATLAS could successfully climb a standard staircase with a 35º incline with ease, owing to its flipper arms giving superior traction. When ATLAS attempted to climb a 45º incline as per the RoboCup challenge, a mechanical failure occurred when one of the flipper motors detached from its mount due to a sheared screw (Figure 50). The cause was unfortunately something that we previously were concerned about, discussed in Section 4.5.2. The old flipper motor, which did not have the capability to reach a bearing fixed to the chassis, collapsed. The torque from the chain when ATLAS was on the steep incline was sufficient to shear one of the older screws and the bracket detached.

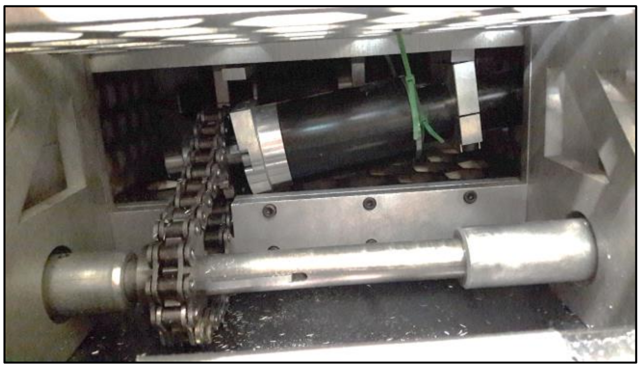


Figure 50: Mechanical Failure – Bracket Detachment

Despite the mechanical failure, ATLAS was successful in all tests that were trailed. This meant that the ATLAS project reached a new benchmark as it surpassed the achievements of the past 3 years in terms of mobility and manoeuvrability. These results will hopefully encourage future sponsorship so that older components can be replaced.

6.2. Exploration Testing

As a rescue robot, it is essential to locate victims and to establish communication. This can be done through capturing thermal image and audio acuity. Image processing ability is also required to identify hazard warning labels.

6.2.1. Testing the Kinect-based SLAM

Calibration was performed on both the depth sensor and RGB camera by moving a checkerboard through 3D space in different orientations and having a ROS program *camera_calibration* [58] handle the video data. This is shown in Figure 51. This sets the focal length and distortion model required for high-accuracy video analysis.

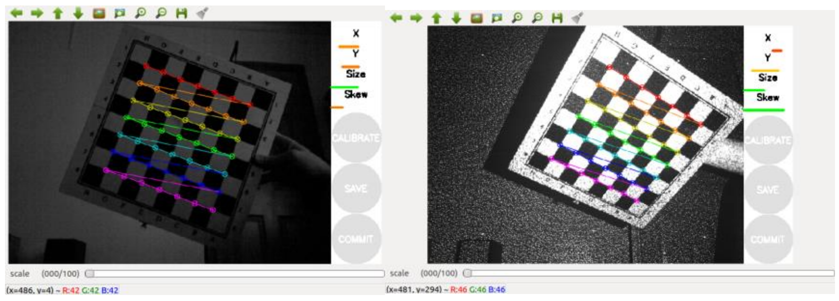


Figure 51: Calibrating the Kinect

The first set of tests for the Kinect-based SLAM were run along a corridor in WMG at the University of Warwick. This was shown in Figure 52 with the left image showing a point-of-view (POV) point cloud of the corridor and the centre image showing an aerial view of the 3D model generated. It was observed that in feature-sparse environments, 3D SLAM becomes extremely challenging with current technology and methods. The curved point clouds of the corridor contrast with the photograph to the right of the figure, showing the straight corridor. Careful consideration for parameters were necessary to ensure successful loop-closure detection and feature extraction, with the carpet pattern likely causing the issues here.

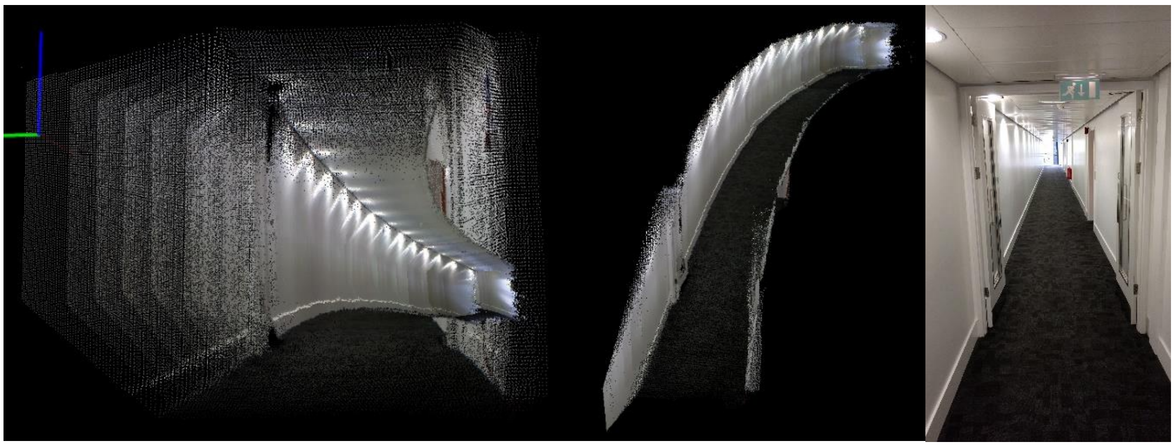


Figure 52: WMG Corridor [from left to right]: Point of View Image, Aerial View image and Photograph

Finally, the performance of Kinect-based SLAM was demonstrated in feature-rich environments in lecture room 114 and the two student syndicate study areas in WMG, as seen in Figure 53.

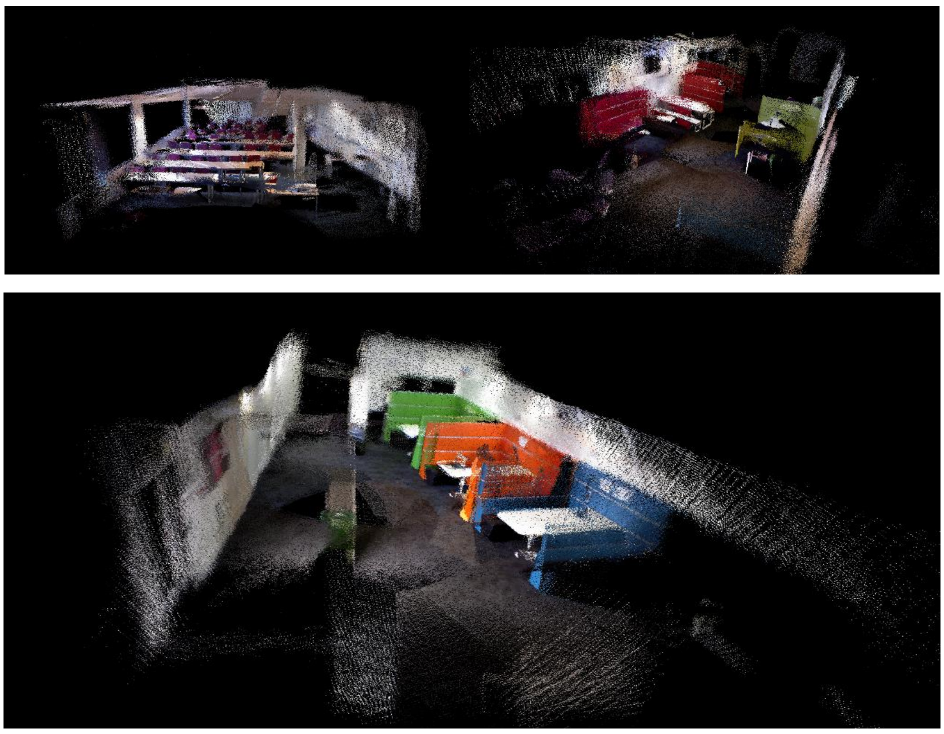


Figure 53: Using the Kinect in feature rich environments

6.2.2. LiDAR Mapping

The mapping ability of this device in buildings was tested along corridors throughout the University of Warwick Engineering department. Shown in Figure 54 are the second, third and fourth floors of the Engineering building.

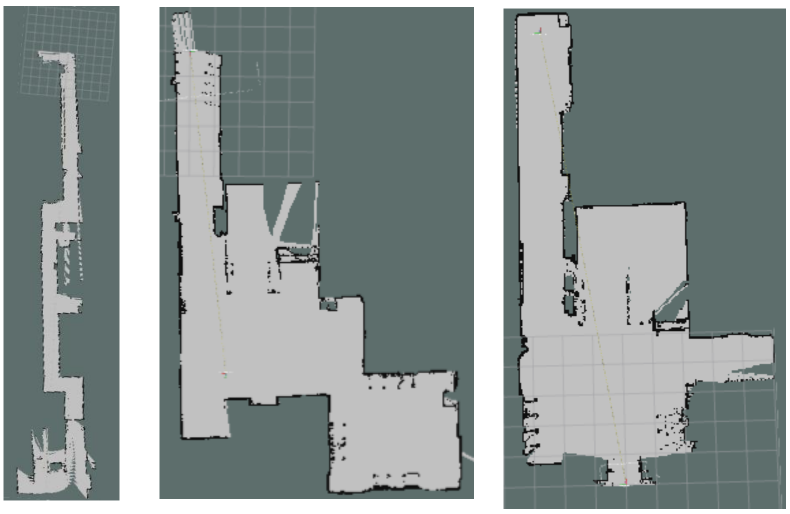


Figure 54: LiDAR maps of Warwick's engineering department (from left to right): second floor, third, floor, fourth floor

The walls of the building are clearly visible and updated at a high enough frequency for even a fast moving robot to navigate through. This meant that the LiDAR system performed to the standard required for the RoboCup tournament [17]. Future years should work towards path planning and object avoidance & targetting.

6.2.3. CO_2 Sensor Testing

The CO_2 sensor was left to heat up for a 24 hour period to allow enough time for the sensing module to anti-oxidise before readings could be taken. Once heated, testing was performed in a standard work room environment.

Figure 55 shows a graph of the results. The sensor was blown on at four distinct points over a 500 second time frame; 50 seconds, 200 seconds, 300 seconds and 400 seconds. The sensitivity of the sensor proved to be high, with the readings spiking within two seconds after the sensor had been blown on.

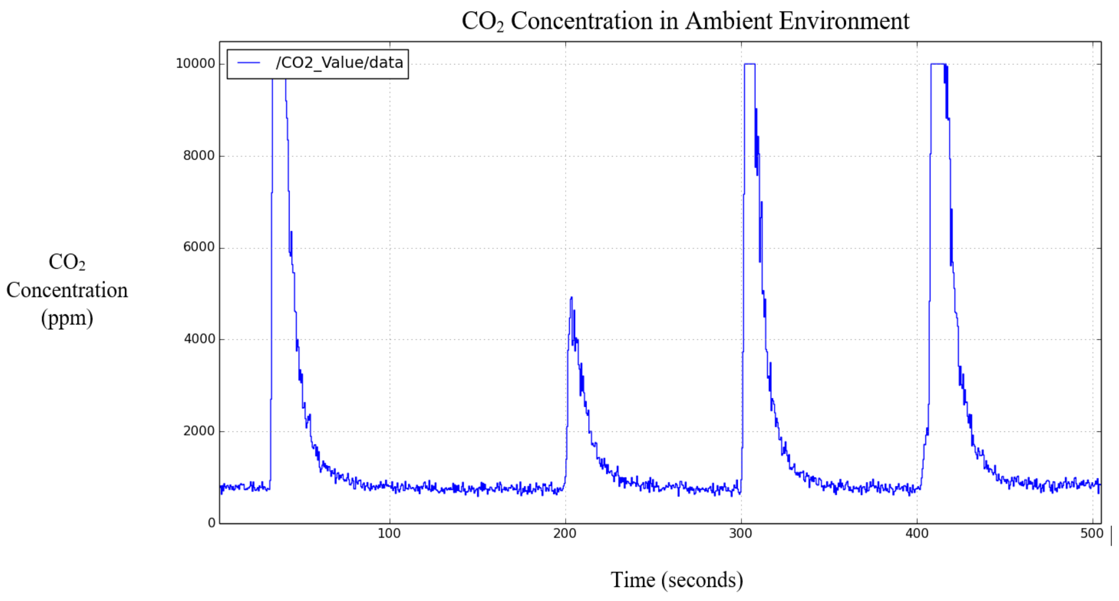


Figure 55: CO_2 Sensor testing: Concentration of CO_2 (ppm) vs Time (seconds)

The steady state readings were found to be a concentration of 700 – 800 ppm. As ambient CO_2 is normally around 400ppm it is possible that further calibration of the sensor is required in the future. A CO_2 meter was not available for more advanced calibration. However, the sensor was deemed suitable to meet the RoboCup requirements of showing an "*active display of increase in CO2 concentration when a teammate breaths into the robot's sensor or a CO2 cartridge is opened near the sensor"* [17]. A flowchart of the code written to read the sensor readings can be found in Appendix A.b along with the sensor's operating theory.

7. Critical Review of ATLAS Project

7.1. 2016/17 WMR Team

A multi-disciplinary team was greatly beneficial to the project as it allowed for a natural split in tasks to be undertaken. Furthermore, good communication and cohesion between the team was achieved via weekly meetings and agendas enabling the sub-groups to maintain a clear picture of the progress being made throughout the year. This additionally meant work was able to be performed simultaneously by the different sub-teams.

A structured and clear approach to the team helped to identify who was in charge and working on various areas. This helped to identify who held responsibilities and should be the first point of contact should any issues arise with different areas of the work.

Having the outlined 10-stage plan aided to give the team an organised process by which to continuously review progress against. One thing that was taken away from this project is the importance of time keeping. Meeting set deadlines is crucial, especially in a co-dependent team project where successive tasks are dependent on certain jobs being completed first.

7.2. Project, Design and Manufacture

As the decision was made to undertake a complete redesign of the robot from the previous year, the team found that there was added manufacturing pressure. The mechanical design that was produced for ATLAS was found to be robust but too heavy. A lack of sponsors meant that there were restricted funds for the team to operate. As a result, executive decisions had to be made about which aspects of the project should be prioritised and thus where the money should be spent. Although proposed and designed months before the deadline, an attempt to get the electronic boards built for the first time within WMG were met with a lot of issues as a result of a machine breakdown and delayed subsequent maintenance. Returning to the old manufacturer (Warwick Engineering) required changing board designs to accommodate the different machining capabilities. This caused further delay times. In the future, more lead time should have been allowed for manufacturing PCB boards to allow for proper testing and subsequent design alterations.

The approach of creating a simpler model was proven to be an effective choice as a functioning robot was produced with potential to build upon and optimise in the future.

8. Conclusion

Past WMR projects were successfully reverse engineered. This provided a solid ground base for which the 2016/17 robot ATLAS was designed. The team utilised components to satisfy minimum requirements as dictated by the functionality phase of the 10-stage plan (stages 3 -6). Working motor control for the inner tracks and flipper arms were successfully achieved. Within the project time frame, the suspension and tensioning designs outlined in the innovation phase of the plan were investigated, proposed and at the prototyping stage by the conclusion of the project. A detailed explanation of this was included in the report to allow for next year's team to follow this through. Compared to the last few WMR teams, the ATLAS team far surpassed the sensing capabilities of Cyclone and Orion, by managing to obtain a working SLAM system, prove QR code reading ability and obtain meaningful CO_2 sensor readings. This also ticks off stage 9 of the plan. Finally, real-world testing was undergone where ATLAS was able to prove both mobility and manoeuvrability by performing basic drive functions as outlined in the RoboCup requirements. Thus, completing stage 10 of the outlined 10-stage plan.

9. Future Recommendations

In addition to the innovations that were suggested in this technical report, this sections details other future recommendations for ATLAS.

- Investigate the potential to use a two-speed gearbox to give more versatile mobility.
- Develop a compact cooling system to prevent overheating of the electronics while the chassis remains fully enclosed to prevent the ingress of dirt and water.
- More sophisticated software should be used to perform drop testing of the chassis.
- Consider the use of alternative high strength lightweight materials for the chassis such as Carbon fibre if funding allows. This may only be possible if financial sponsorship is increased.
- Consider anodizing exposed parts like bearings and flippers to improve resistance to corrosion.
- Consider linking the IMU to the motor controller to prevent any high accelerations being applied to the robot on steep inclines.
- Consider implementing additional batteries. Potentially four, with two in series and then those two in parallel to provide a separate steady power source for the flippers and inner tracks.
- A hardware emergency stop should be integrated on the board along with a clear LED array for status indication on-board ATLAS.
- Include encoders to allow for more accurate monitoring of the orientation and speed of the motors.
- Acquire a new flipper motor to prevent the mechanical failure re-occuring.
- Make use of the new battery monitoring capability and investigate potential ways to balance the battery cells to allow for a more uniform discharge using the LTC6802 chip.
- Assimilate the infrared camera on to the sensor array. New connection cables will need to be acquired.
- Develop a robot arm to include the sensors and a gripping end-effector.
- Create an rqt GUI interface integrated in ROS to toggle Boolean topics and be more easily reconfigurable.
- Include microphone and speaker to allow for communication from the base station to those in the vicinity of ATLAS.

10. References

[1] A. Davids, “"Urban Search and Rescue Robots: From Tragedy to Technology",” *IEE Intelligent Systems,* 2002.

[2] W. M. Robotics, “Technical Report,” 2013.

[3] Warwick Mobile Robotics, “ATLAS: Cost Benefit Analysis,” 2016/17.

[4] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset and A. M. Erkmen, “Search and Rescue Robotics,” in *Springer Handbook of Robotics* , B. Siciliano and O. Khatib, Eds., Springer Berlin Heidelberg, 2008, pp. 1151-1173.

[5] A. Kleiner, F. Heintz and S. Tadokoro, “Editorial: Safety, Security, and Rescue Robotics,” *Journal of Field Robotics,* vol. 33, no. 3, pp. 263-264, 2016.

[6] S. H. Schneider, S. Semenov, A. Patwardhan, I. Burton, C. H. D. Magadza, M. Oppenheimer, A. B. Pittock, A. Rahman, J. B. Smith, A. Suarez and F. Yamin, “Assessing Key Vulnerabilities and the Risk from Climate Change,” in *Climate Change 2007: Impacts, Adaptation and Vulnerability,* Cambridge, Cambridge University Press, 2007, pp. 779-810.

[7] S. Tadokoro, Rescue Robotics: DDT Project on Robots and Systems for Urban Search and Rescue, London: Springer-Verlag London, 2009.

[8] C. Boyette, “Robots, drones and heart-detectors: How disaster technology is saving lives,” 5 October 2015. [Online]. Available: http://edition.cnn.com/2015/08/24/us/robot-disaster-technology/.

[9] “Fukushima Accident 2011,” 2017 January. [Online]. Available: http://www.world-nuclear.org/info/Safety-and-Security/Safety-of-Plants/Fukushima-Accident-2011/#.UW_2yit37l2. [Accessed 2 April 2017].

[10] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake and K. Yoshida, “Emergency response to the nuclear accident at the Fukushima Daiichi nuclear power plants using mobile rescue robots,” *Journal of Field Robotics,* vol. 30, no. 1, pp. 44-63, 2012.

[11] “Quince,” 2011. [Online]. Available: http://furo.org/en/works/quince.html. [Accessed 15 April 2017].

[12] B. Siciliano and O. Khatib, Handbook of Robotics, Berlin: Springer Science and Business Media, 2008.

[13] M. Rossi and B. Davide, “Autonomous Gas Detection And Mapping With Unmanned Aerial Vehicles,” *IEEE Transactions on Instrumentation & Measurement,* pp. 1-11, 2015.

[14] T. Ngoma, *Limitations of current robots outside the factory: Search and Rescue,* 2016.

[15] S. Chakraborty and S. Basu, “Design of a Smart Unmanned Ground Vehicle for Hazardous Environments,” in *2nd National Conference on Recent Trends in Information Systems*, Kolkata, 2008.

[16] Robot League Wiki, “Rescue Robot League Wiki,” 12 4 2017. [Online]. Available: http://wiki.robocup.org/Robot_League#RoboCup_2017_Information.

[17] RoboCup, “RoboCup Rescue Rulebook,” 17 6 2016. [Online]. Available: http://wiki.robocup.org/images/6/65/2016-03RoboCupRescueRulebookv3.pdf.

[18] GRM Consulting, “Genesis Software,” GRM, Royal Leamington Spa, 2017.

[19] SAPA, “Matweb.com,” 2001. [Online]. Available: http://www.matweb.com/search/datasheet_print.aspx?matguid=fad29be6e64d4e95a241690f1f6e1eb7 . [Accessed 9th January 2017].

[20] W. Bauer, hydorpneumatic Suspension Systems, New York: Springer, 2011.

[21] U.S. Department of the Army, Washington D.C.: U.S. Government Printing Office, 1947.

[22] P. Horowitz, The Art of Electronics, New York: Cambridge University Press, 2015.

[23] S. Ericson, M. Kelly, N. Marshall, R. Navarro, C. Newton, J. Parkhurst, A. Pranaitis and R. Waldron, “Autonomous Unmanned Aerial Vehicle,” in *Preparation of Papers for AIAA Technical Conferences*, Maryland, 2014.

[24] B. Witherspoon, “Voltage Regulation using Zener Diodes,” Michigan State University: College of Engineering , Michigan, 2008.

[25] C. Simpson, “Linear and Switching Voltage Regulator Fundamental Part 1,” National Semiconductor, Texus, 2011.

[26] E. Zauls, “WMR 12/13 Reports,” May 2013. [Online]. Available: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/projects/rescue/reports/1213reports/.

[27] E. Rushford, “Search and Rescue: Technical Reports,” April 2017. [Online]. Available: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/projects/rescue/reports/1415reports-copy/wmr_technical_report_2014-15.pdf.

[28] TT Erectronics, “RESISTORS FOR CIRCUIT PROTECTION - Application Note,” n.d. [Online]. Available: http://www.ttelectronics.com/themes/ttelectronics/datasheets/resistors/literature/Circuit-Protection_AN.pdf.

[29] J. Salt, “Understanding RC LiPo Batteries,” 2017. [Online]. Available: http://www.rchelicopterfun.com/rc-lipo-batteries.html. [Accessed 6 April 2017].

[30] I. Advanced Energy, *Rechargeable Lithium Ion Polymer Battery*, Las Vegas, Nevada, 2011.

[31] WMR, *Lithium Polymer Battery Risk Assessment*, 2010.

[32] B. Schneider, “A Guide to Understanding Lipo Batteries,” 24 March 2017. [Online]. Available: https://rogershobbycenter.com/lipoguide/. [Accessed 8 April 2017].

[33] WMR, “Next Generation Urban Search and Rescue Robotics,” 2016.

[34] TURNIGY, “Turnigy nano-tech 6000mah 6S 25~50C Lipo Pack,” 2017. [Online]. Available: https://hobbyking.com/en_us/turnigy-nano-tech-6000mah-6s-25-50c-lipo-pack.html. [Accessed 6 April 2017].

[35] Linear Technology, “LTC6802-2 Multicell Addressable Battery Stack Monitor,” 2017. [Online]. Available: http://cds.linear.com/docs/en/datasheet/68022fa.pdf.

[36] “Arduino Blog | Battery Management System (LTC6902 & Arduino),” 20 July 2014. [Online]. Available: https://sourcelion.wordpress.com/2014/07/20/battery-management-system-ltc6802-arduino/. [Accessed 6 April 2017].

[37] Arduino, “SPI library,” 2017. [Online]. Available: https://www.arduino.cc/en/reference/SPI. [Accessed 6 April 2017].

[38] B. Crofts and K. Godfrey, “Basic Electric Circuits and Components,” in *Circuits and Systems*, Harlow, Pearson Education Limited, 2009, pp. 26-27.

[39] S. Hill, “Measuring Currnet To Detect Out-of-Range Conditions,” July 2016. [Online]. Available: http://www.ti.com/lit/an/sboa162a/sboa162a.pdf. [Accessed April 2017].

[40] J. Flannery, H. Francis, M. Gloger, A. Lamm, Y.-Y. Lau and D. Riley, “Warwick Mobile Robotics: Search and Rescue,” May 2016. [Online]. Available: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/projects/rescue/reports/1516reports-copy/submission_tech_report.pdf. [Accessed April 2017].

[41] Maxon Motors, “ESCON 50/50: Hardware Reference,” November 2015. [Online]. Available: http://www.maxonmotor.co.uk/medias/sys_master/root/8818447482910/409510-ESCON-50-5-Hardware-Reference-En.pdf. [Accessed April 2017].

[42] Square D: Schneider Electric, “Circuit Breaker Characteristic Trip Curves and Coordination,” August 2001. [Online]. Available: http://static.schneider-electric.us/docs/Circuit%20Protection/Molded%20Case%20Circuit%20Breakers/0100-400%20A%20Frame%20FA-LA/FA-FC-FH/0600DB0105.pdf. [Accessed April 2017].

[43] Texus Instruments, “TI Over-Current Detection Products,” 2014. [Online]. Available: http://www.ti.com/lit/ml/slob091/slob091.pdf. [Accessed 2017].

[44] Intel Support, “Different Wi-Fi Protocols and Data Rates,” 4 2017. [Online]. Available: http://www.intel.co.uk/content/www/uk/en/support/network-and-i-o/wireless-networking/000005725.html.

[45] K. AMERASINGHE, “H.264 FOR THE REST OF US,” n.d. [Online]. Available: http://www.adobe.com/content/dam/Adobe/en/devnet/video/articles/h264_primer/h264_primer.pdf.

[46] "About ROS," 2017. [Online]. Available: http://www.ros.org/about-ros/. [Accessed 12 April 2017].

[47] E. Fernandez, L. S. Crespo, A. Mahtani and A. Martinez, Learning ROS for Robotics Programming, 2nd ed., Birmingham: Packt Publishing, 2015.

[48] WMR, "Urban Search & Rescue: Design & Development of a Miniature, Urban Search & Rescue Robot," 2015.

[49] MAXON MOTOR, "ESCON 50/5 Hardware Reference," September 2013. [Online]. Available: http://www.maxonmotor.com/medias/sys_master/8810873815070/409510-ESCON-50-5-Hardware-Reference-En.pdf.

[50] DimensionEngineering, "Sabertooth dual 25A motor driver," July 2007. [Online]. Available: https://www.dimensionengineering.com/products/sabertooth2x25. [Accessed 13 April 2017].

[51] Ion Motion Control, "RoboClaw Series Brushed DC Motor Controllers," 2016. [Online]. Available: http://downloads.ionmc.com/docs/roboclaw_user_manual.pdf.

[52] CORONA LABS, "WHY CHOOSE CORONA ?," n.d. [Online]. Available: https://coronalabs.com/corona-sdk/.

[53] D. Thomas, "ROS Package Summary," 8 2016. [Online]. Available: http://wiki.ros.org/rqt.

[54] Python Software Foundation, "Low-level networking interface," 27 March 2017. [Online]. Available: https://docs.python.org/2/library/socket.html. [Accessed 24 April 2017].

[55] Python Software Foundation, "Simple HTTP request handler," 27 March 2017. [Online]. Available: https://docs.python.org/2/library/simplehttpserver.html. [Accessed 14 April 2017].

[56] ROS, "ROS - usb_cam," 05 March 2016. [Online]. Available: http://wiki.ros.org/usb_cam. [Accessed 1 May 2017].

[57] ROS, "ROS - web_video_server," 27 March 2016. [Online]. Available: http://wiki.ros.org/web_video_server. [Accessed 27 April 2017].

[58] ROS, "ROS - openni_launch/Tutorials/IntrinsicCalibration," 06 February 2015. [Online]. Available: http://wiki.ros.org/openni_launch/Tutorials/IntrinsicCalibration. [Accessed 15 March 2017].

[59] J. Suarez and R. R. Murphy, "Using the Kinect for Search and Rescue Robotics," in *IEEE International Symposium on Safety, Security, and Rescue Robotics,* 2012.

[60] ROS, "ROS - rtabmap," 22 February 2015. [Online]. Available: http://wiki.ros.org/rtabmap. [Accessed 26 March 2017].

[61] W. Shang, X. Cao, H. Ma and H. W. P. Zang, “Kinect-Based Vision System of Mine Rescue Robot for Low Illuminous Environment,” *Journal of Sensors,* vol. 2016, p. 9, 2015.

[62] Robot Globe, “Now Kinect 3D Camera can also be used in Bright Sunlight for outdoor Applications,” 2017. [Online]. Available: http://robotglobe.org/now-kinect-3d-camera-can-also-be-used-in-bright-sunlight-for-outdoor-applications/. [Accessed 07 April 2017].

[63] “Earth's CO2 Home Page,” April 2017. [Online]. Available: https://www.co2.earth/. [Accessed 9 April 2017].

[64] NASA, “Carbon Dioxide,” April 2017. [Online]. Available: https://climate.nasa.gov/vital-signs/carbon-dioxide/. [Accessed 9 April 2017].

[65] C. Meter, “What is Carbon Dioxide?,” December 2013. [Online]. Available: https://www.co2meter.com/blogs/news/10709101-what-is-carbon-dioxide. [Accessed 9 April 2017].

[66] dfrobot, “CO2 Sensor SKU:SEN0159,” 5 April 2017. [Online]. Available: http://www.dfrobot.com/wiki/index.php/CO2_Sensor_SKU:SEN0159. [Accessed 9 April 2017].

[67] Robocup, “RoboCup Rescue Rulebook,” RoboCup Wiki, 2016.

[68] Futurlec, “CO2SENSOR - Carbon Dioxide Sensor,” 2017. [Online]. Available: http://www.futurlec.com/CO2_Sensor.shtml. [Accessed 9 April 2017].

[69] D. Nosowitz, “Meet Japan's Earthquake Search-and-Rescue Robots,” 11 March 2011. [Online]. Available: http://www.popsci.com/technology/article/2011-03/six-robots-could-shape-future-earthquake-search-and-rescue.

[70] R. R. Murphy, “Trial by fire [rescue robots],” *IEEE Robotics & Automation Magazine,* vol. 11, no. 3, pp. 50-61, 2004.

[71] A. Chapman, “Types of Drones: Multi-Rotor vs Fixed-Wing vs Single Rotor vs Hybrid VTOL,” 8 January 2016. [Online]. Available: http://www.auav.com.au/articles/drone-types/. [Accessed 18 April 2017].

[72] R. Bogue, “Robots in the nuclear industry: a review of technologies and applications,” *Industrial Robot: An International Journal,* vol. 38, no. 2, pp. 113-118, 2011.

[73] “Daksh Remotely Operated Vehicle (ROV), India,” 11 April 2017. [Online]. Available: http://www.army-technology.com/projects/remotely-operated-vehicle-rov-daksh/.

[74] P. Vis, “Heat Sink Voltage Regulator 7805,” [Online]. Available: http://www.petervis.com/electronics/Voltage_Regulator_Heatsink/Heatsink_for_TO-220_Voltage_Regulator.html. [Accessed 6 April 2017].

[75] TEXAS INSTRUMENTS, “SN74HC11 Triple 3-Input Positive-AND Gate,” August 2003. [Online]. Available: http://www.ti.com/lit/ds/symlink/sn74hc11.pdf.

[76] TEXAS INSTRUMENTS, “A7800 Series Positive-Voltage Regulators,” May 2003. [Online]. Available: https://www.sparkfun.com/datasheets/Components/LM7805.pdf.

[77] ROS, “Writing a Teleoperation Node for a Linux-Supported Joystick,” 20 November 2016. [Online]. Available: http://wiki.ros.org/joy/Tutorials/WritingTeleopNode. [Accessed 11 April 2017].

[78] ROS, “ROS/StartGuide,” 5 April 2017. [Online]. Available: http://wiki.ros.org/ROS/StartGuide. [Accessed 11 April 2017].

[79] D. Mohankumar, “Battery Level Monitor,” [Online]. Available: http://www.electroschematics.com/6338/battery-level-monitor/. [Accessed 6 April 2017].

[80] D. Lancaster, CMOS Cookbook, 2nd ed., Newnes, 1997.

[81] “Batteries for UAV,” 2017. [Online]. Available: http://dronesarefun.com/BatteriesForUAV.html. [Accessed 6 April 2017].

[82] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, Principles of robot motion: theory, algorithms, and implementation, A Bradford Book, 2007, p. 302.

[83] M. C. Orlowski, “Defense Advanced Research Projects Agency,” 2015. [Online]. Available: http://www.darpa.mil/program/darpa-robotics-challenge.

[84] Robot League, “Test reference”.

[85] Corona Labs, “Why Choose Corona?,” Corona Labs, 2017. [Online]. Available: https://coronalabs.com/corona-sdk/. [Accessed 30 April 2017].

[86] ROS, “ROS - rqt,” 03 08 2016. [Online]. Available: http://wiki.ros.org/rqt. [Accessed 28 April 2017].

[87] “Simple HTTP request handler,” Python Software Foundation, 27 March 2017. [Online]. Available: https://docs.python.org/2/library/simplehttpserver.html. [Accessed 18 April 2017].

[88] E. Schleicher, “GitHub - introlab/rtabmap,” 04 February 2017. [Online]. Available: https://github.com/introlab/rtabmap/wiki/IROS-2014- Kinect-Challenge. [Accessed 09 March 2017].

11. Appendices

Appendix A. Gantt Chart

		DUE DATE						
		Week 25	Week 26	Week 27	Week 28	Week 29	Week 30	Week 31
		20/03/2017	27/03/2017	03/04/2017	10/04/2017	17/04/2017	24/04/2017	01/05/2017
	Static Tensioning Design	🟩	🟩					
	Dynamic Tensioning & Calculations		🟩	🟩				
	Cladding Design & Manufacture			🟩	🟩			
	Strategy/Composition + Aims/Obj.				🟩	🟩		
Oliver	Optimised Chassis FEA + Presentation					🟩	🟩	🟩
	Basic Mechanical Structure	🟧	🟧					
	Solid and Dynamic Suspension		🟧	🟧	🟧			
	Supply Drop / Jockey Wheel				🟧	🟧		
	Design for Manufacture					🟧	🟧	
Dan	Real-World Mechanical Testing						🟧	🟧
	Critical Analysis of Champion	🟥	🟥	🟥				
	3D Slam Kinect + LiDAR			🟥	🟥			
	Mobile Device App				🟥	🟥	🟥	
	Image Rendering for Report					🟥	🟥	🟥
Andrew	Real-World Mechanical Testing						🟥	🟥
	Power Lit. Review	🟦	🟦	🟦				
	Cost Benefit Analysis	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	Current Sensing and Protection			🟦	🟦			
	Current Protection of Power Sys.					🟦	🟦	
Guy								
	Motivation - USAR Lit. Review	🟧	🟧					
	Motor Control / Drive System		🟧	🟧				
	C02 & IMU			🟧	🟧			
	ROS Programming				🟧	🟧		
	Battery Monitoring					🟧	🟧	
Thandi	Real-World Electrical Testing						🟧	🟧
	Motivation - WMR & Competitions	🟨	🟨					
	Future Work - Electronics		🟨	🟨				
	System Diagram			🟨	🟨			
	Power Board Design					🟨	🟨	
Adam	Communications / Testing						🟨	🟨
	Flipper Arm Redesign	🟪	🟪	🟪				
	Centre of Mass / Climbing Calc.			🟪	🟪			
	Thermal Analysis / Fan Positioning				🟪	🟪		
	Electronics Packaging / Layout					🟪	🟪	
Michael	Optimise Drop Testing					🟪	🟪	
DEADLINE								
Thursday 4th May								

Appendix B. Agenda Example

Week 18 Friday 27th January 2017

Weekly Meeting Agenda

Meeting Format

1) Individual and sub-team review and status of work completed during the week.
2) Discussion of weekly topics proposed by the project manager and resolution of issues that may have arisen.
3) Delegation of tasks and division into smaller sub-teams for the next week of work.
4) Targets that should be met before next week's meeting.

Matters for Discussion

- Schedule Meetings
  - Reschedule weekly meeting
  - Create a team meeting for early next week
- Open days every Wednesday
  - This secures £50 each week for our project (that's almost an idler wheel every 2 weeks!)
  - Decide who is present on each day
- Deadline Day
  - Review and evaluate completion of tasks set previously
- Delegation of New Work
  - To be done on the board in Emma's office and sent to the group's page.
  - Due in for 17th February (in 3 weeks)
  - Consider what needs to be done
    - 10 pages each; your OWN work
    - Needs to be written towards the end of this term

Delegation of Work

Electrical Team	Mechanical Team
Adam	Dan
Guy	Michael
Thandi	Ollie
Andrew	

- Work will be delegated in the meeting. Details will be posted on the agenda for week 19.
  - Deadline is week 6 of the second term (Week 21)

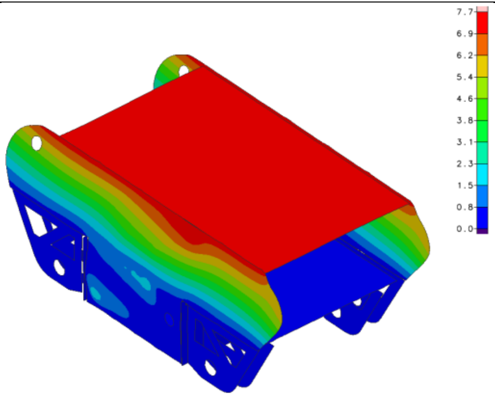
Disclaimer: Naturally these sub-teams are not set in stone and the sub-team members are free to assist other sub-teams at their behest, so long as the weekly targets are met and they can justify the work they have done at the next meeting.

Warwick Mobile Robotics (WMR)

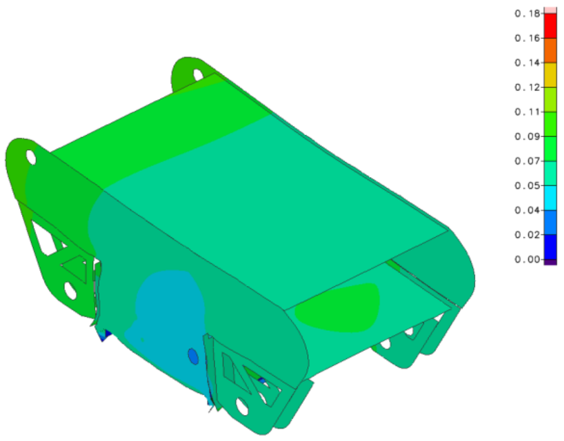
Appendix C. FEA and Topology Results

Stiffness Results

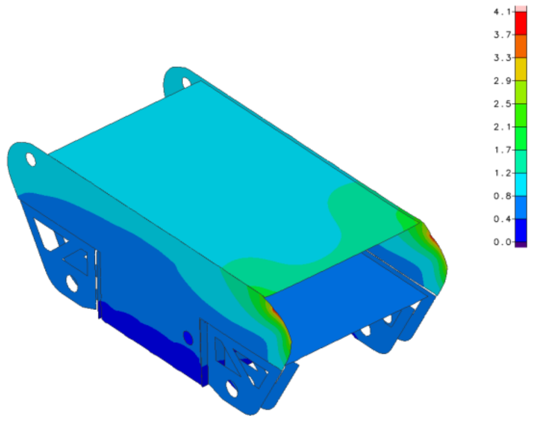
Bottom Impact



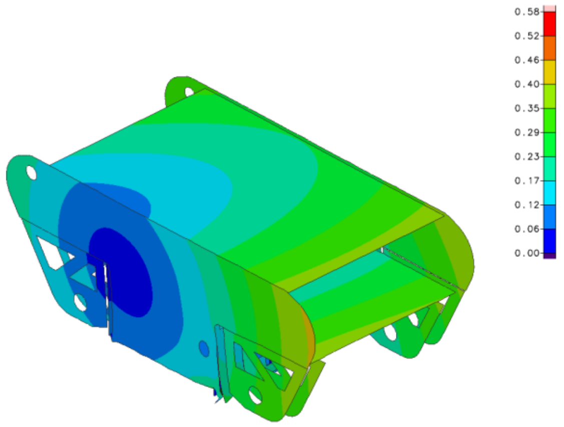
Side Impact



Nose Impact

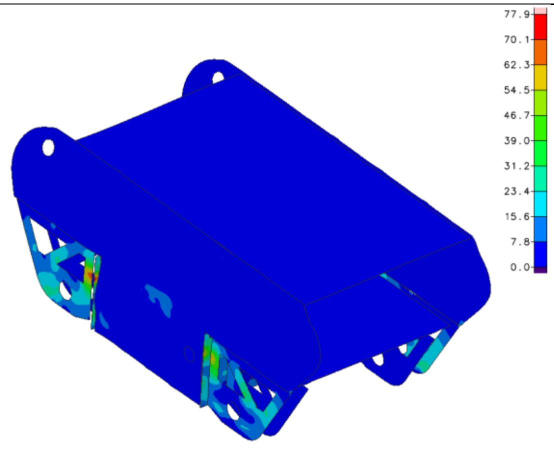


Corner Impact

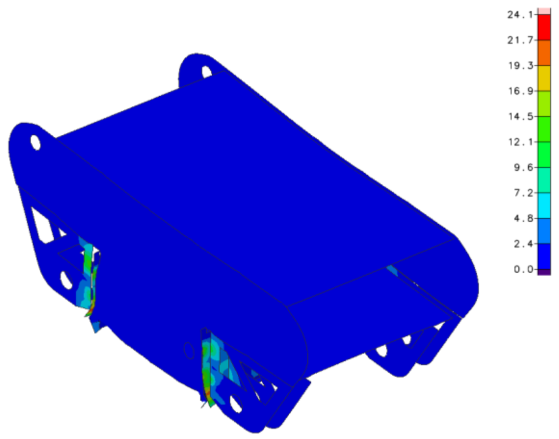


Stress Results

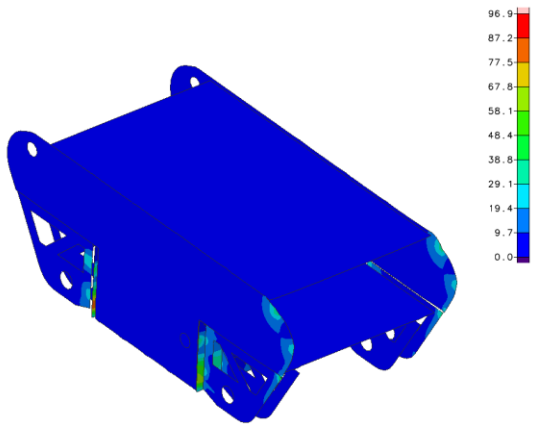
Bottom Impact



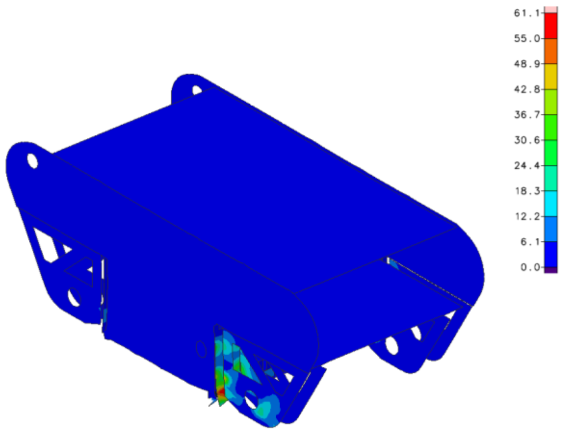
Side Impact



Nose Impact

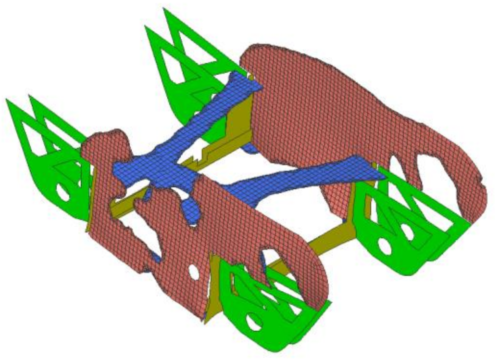


Corner Impact

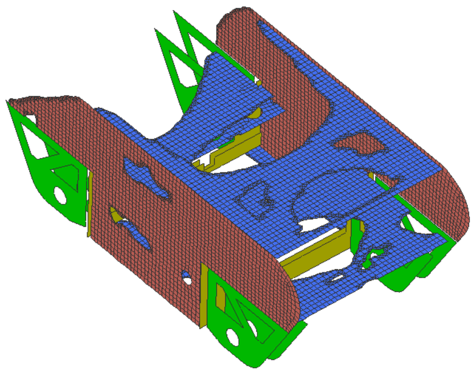


Topology Results

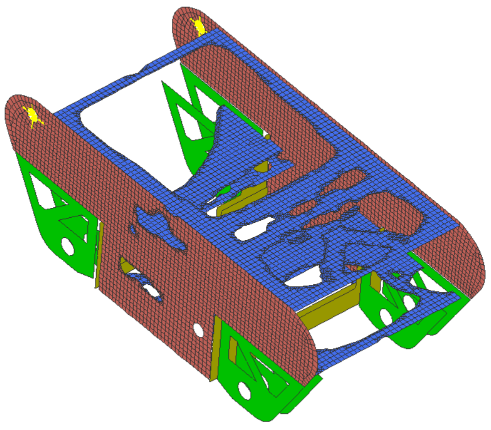
Variant 2



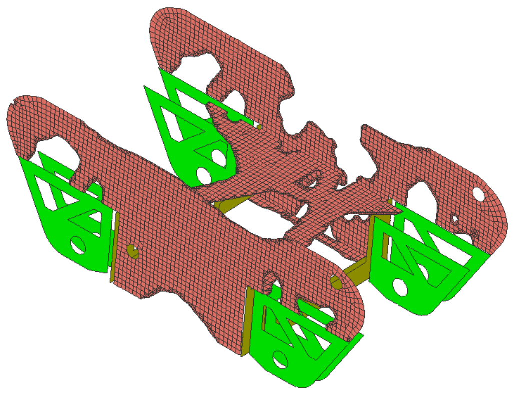
Variant 3



Variant 4



Variant 5



Appendix D. Sensors

a. Xbox Kinect

In situations where data transmission bandwidth is reduced, live video feed becomes difficult. In such cases it could be prudent to post-process the video data to send only the most useful information back to the robot operator. Indeed, when operating autonomously a robot must be able to gain an understanding of its environment from its sensors and on-board processing. This year, Atlas robot has joined a growing trend in mobile robotics [59] to make use of the Xbox Kinect (Figure 56) to efficiently map 3D environments without expensive hardware.

Figure 56: Xbox Kinect

The Xbox Kinect hardware consists of an infra-red depth scanner and an RGB (red, green and blue) camera. Once calibrated, the Kinect is used to achieve RGB-D SLAM(simultaneous localisation and mapping) by employing a ROS wrapper of a real-time appearance-based mapping solution called RTAB-Map [60]This novel technique combines existing SLAM methods with a global loop closure detection to detect previously visited points in space and remap it to the existing map in memory. This solves the “kidnapped robot problem” which is where a robot does not know its initial arbitrary position, a problem exemplified in search and rescue robotics where the environment is changing or recently changed by natural disasters and thus unfamiliar to the operator.

RTAB-Map overlays the depth sensor data with coloured pixels from the RGB camera to create a point cloud visualisation of the scanned environment. In other modes, it combines data from two stereo cameras to achieve similar results. The loop-closure detection runs uses just the RGB camera data to detect “features” previously visited. Features are areas of high-contrast that are tracked in real-time to give the software points in 3D space to relate to the existing (and developing) model of the environment. The left image in Figure 57 shows a feature-rich classroom environment, ideal for this SLAM tracking application. The centre image shows features passing the loop-closure detection. Lastly the right image shows a view in a corridor with few contrasting areas and causing the software to lose tracking ability completely.

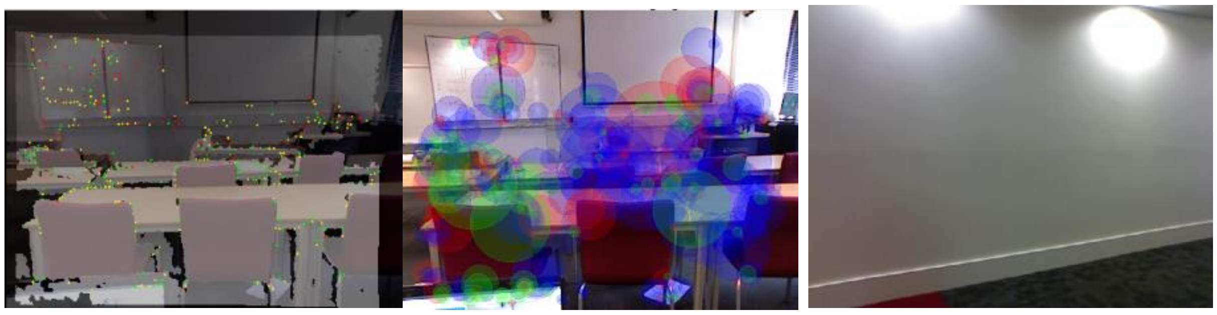

(a) (b) (c)

Figure 57Kinect SLAM: Classroom tracked in real-time (a), Classroom through SLAM closed loop (b), View of corridor (c)

It is expected these 3D point clouds of environments can help the robot operator gain an understanding of the environment when 2D video cannot, as shown with a mine rescue robot [61]. RTAB-Map allows the maps to be created in real-time and retroactively using data collected by the Kinect. This is practical when data bandwidth is reduced. Future years are expected to make use of this state-of-the-art technology to aid in the RoboCup competitions. Despite the Kinect being notoriously poor-performing in outdoor environments, some promise has been shown by limiting extraneous "noisy" light to make the Kinect useful outdoors [62]. This should be explored by future years of the project to expand Atlas's utility.

b. CO_2 Sensing

The core idea behind having a CO_2 sensor on ATLAS is to be able to detect the presence of any potential victims whilst the robot is doing surveillance of an area by looking out for any spikes in CO_2 concentrations. Ambient CO_2 concentrations have been found to be just over 400ppm [63] [64], whereas the normal quantity released during human respiration has been found to be at a much higher concentration of around 40000ppm [65].

The SKU:SEN0159 CO_2 sensor is a simple sensor with an operating range from 400-10000ppm [66]. This particular sensor is more appropriate for qualitative analysis to inform as the gas concentration increases and indicate when it is above the maximum value to be read; in this case, HIGH.

For the RoboCup, it is required to prove an increase in CO_2 concentration when a team-mate breaths over the sensor or a CO_2 cartridge is opened near the sensor [67]. The CO_2 sensor is able to accommodate this by giving increasing readings up until 10000ppm and then sending a signal of HIGH to indicate the concentration has surpassed its maximum threshold limit.

The basic operating principle for this CO_2 Sensor is that the output voltage drops as the CO_2 concentration increases [68]. So the sensor can be calibrated using the on board potentiometer to set the threshold voltage. If the gas concentration is high enough than the voltage should always remain lower than the pre-set threshold. The MG-811 sensor module on board the sensor is extremely sensitive and selective towards CO_2. Figure 58 shows how the induced Electromotive Force (EMF) varies with CO_2 concentration.

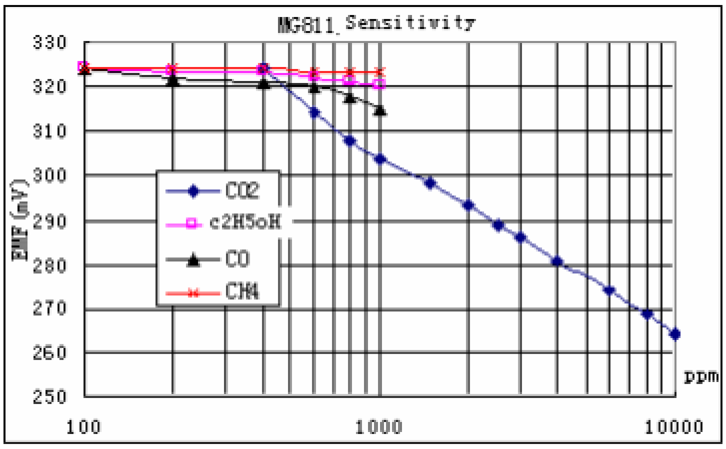


Figure 58*:* MG-811 Sensor Module Gas Sensitivity Curve [68]

The on-board heating circuit brings the module to its optimum operating temperature but requires a 6V heating voltage. An additional 9V power supply to connect to the Arduino Mega was necessary to ensure that the sensor could get enough heating power.

A CO2 curve is created using the format {x, y, slope}. The defined start point on the x-axis of the curve is set to log(400) and the y point is the output voltage at a concentration of 400ppm. The slope is calculated as the reaction voltage (i.e. the voltage drop from moving the sensor from air in to a 10 000ppm CO_2 concentration) divided by the change in x value (log400 – log10000). The sensor is calibrated by determining the output voltages for when in 400ppm, in 10000ppm and the resulting voltage drop.

The script written for the CO_2 is then comprised of three sections. The first section ‘SensorRead()’ reads the a sample of analogue input voltages from the sensor and finds the average voltage. The next function ‘VolttoPercentage()’takes the found voltage average and divides it by a pre-set gain. The value is checked to see if it is within the limits of the maximum and minimum voltages at 10000ppm and 400ppm respectively. If it is then the gas concentration in ppm is read off of the defined CO_2 curve and printed. If it is above the maximum voltage than a signal HIGH is sent. Finally, in the main loop, the voltage value and concentration percentages are published in ROS on the topic ‘CO2value’.

The flowchart below shows the principle behind how the CO_2 Sensing code works.

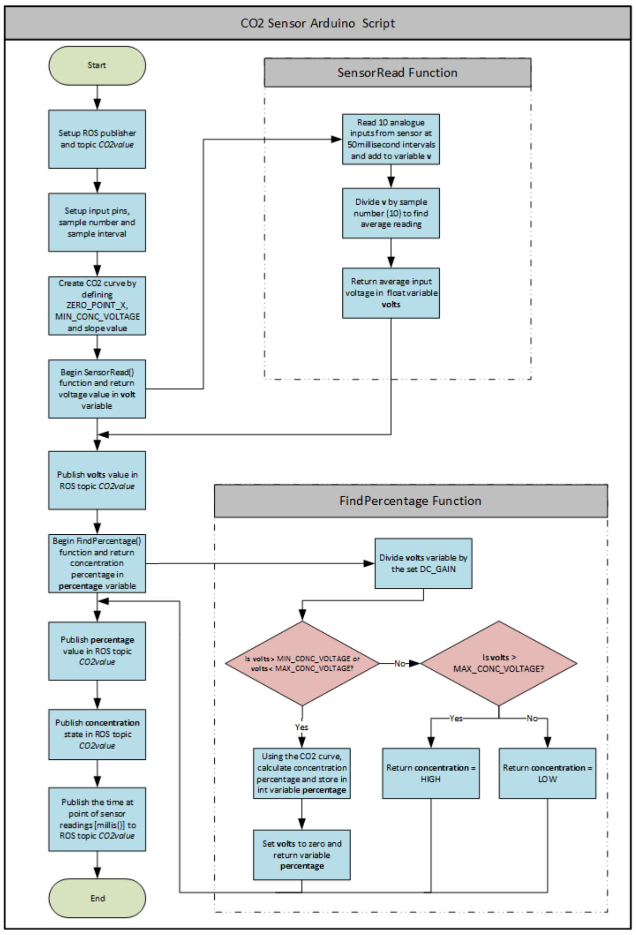
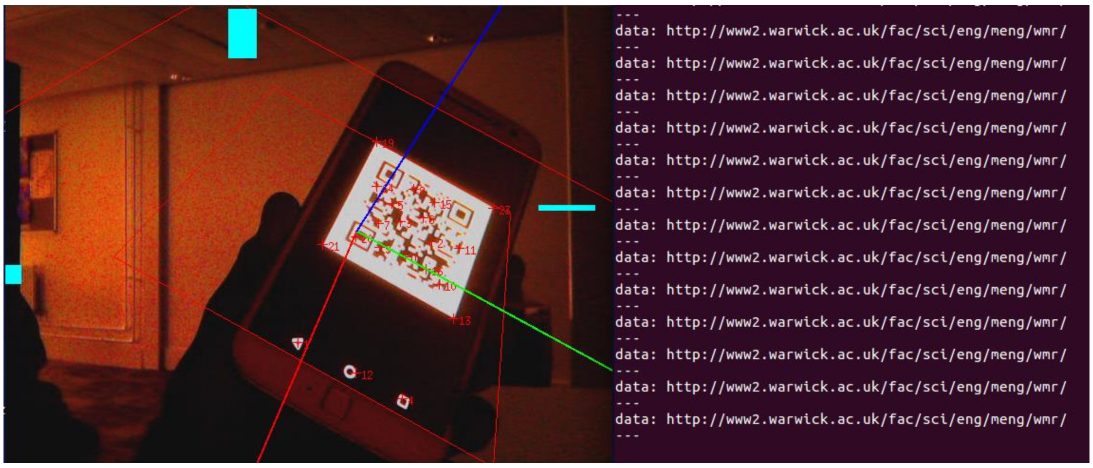

CO2 Sensor Arduino Script
Start
Setup ROS publisher and topic *CO2value*
Setup input pins, sample number and sample interval
Create CO2 curve by defining ZERO_POINT_X, MIN_CONC_VOLTAGE and slope value
Begin SensorRead() function and return voltage value in **volt** variable
Publish **volts** value in ROS topic *CO2value*
Begin FindPercentage() function and return concentration percentage in **percentage** variable
Publish **percentage** value in ROS topic *CO2value*
Publish **concentration** state in ROS topic *CO2value*
Publish the time at point of sensor readings [millis()] to ROS topic *CO2value*
End
SensorRead Function
Read 10 analogue inputs from sensor at 50millisecond intervals and add to variable **v**
Divide **v** by sample number (10) to find average reading
Return average input voltage in float variable **volts**
FindPercentage Function
Divide **volts** variable by the set DC_GAIN
Is volts > MIN_CONC_VOLTAGE or volts < MAX_CONC_VOLTAGE?
No
Is **volts** > MAX_CONC_VOLTAGE?
Yes
Using the CO2 curve, calculate concentration percentage and store in int variable **percentage**
Set **volts** to zero and return variable **percentage**
Yes
No
Return **concentration** = HIGH
Return **concentration** = LOW

Figure 59*:* CO2 Sensor Flowchart

c. QR Code

It is a requirement of the RoboCup rescue league to have automatic QR code recognition [17]. Using the flexible ROS package *visp_auto_tracker*, this has been achieved. The software runs at frame-rate speed and requires no input or direction from the operator. A demonstration of its application is shown in Figure 60



```
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
data: http://www2.warwick.ac.uk/fac/sci/eng/meng/wmr/
---
```

Figure 60 QR Code Operating Demonstration

Appendix E. Light Control

Using a Vishay 4n34 Dip6 Optocoupler, it was possible to hack the push button on a bicycle light. With easy mounting, three 3800 lumens (total) CREE LEDs and a 4800mAh battery pack, this light is ideal for use in search and rescue operations. It is programmed to have four modes: low power, medium power, high power and strobe. Pin 1 on the optocoupler is connected to an Arduino digital output in series with a 220Ω resistor, as shown in figure Figure 61. The light is easily controllable with the controller through ROS nodes.

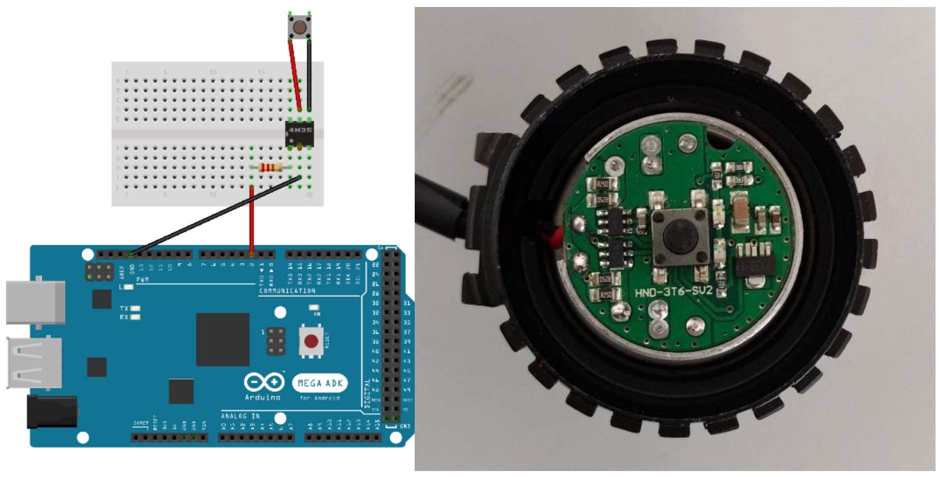


Figure 61 Fritzing Diagram (Left) and the opened rear cover of the headlight (Right)

Appendix F. Power Distribution Circuit Schematic and PCB

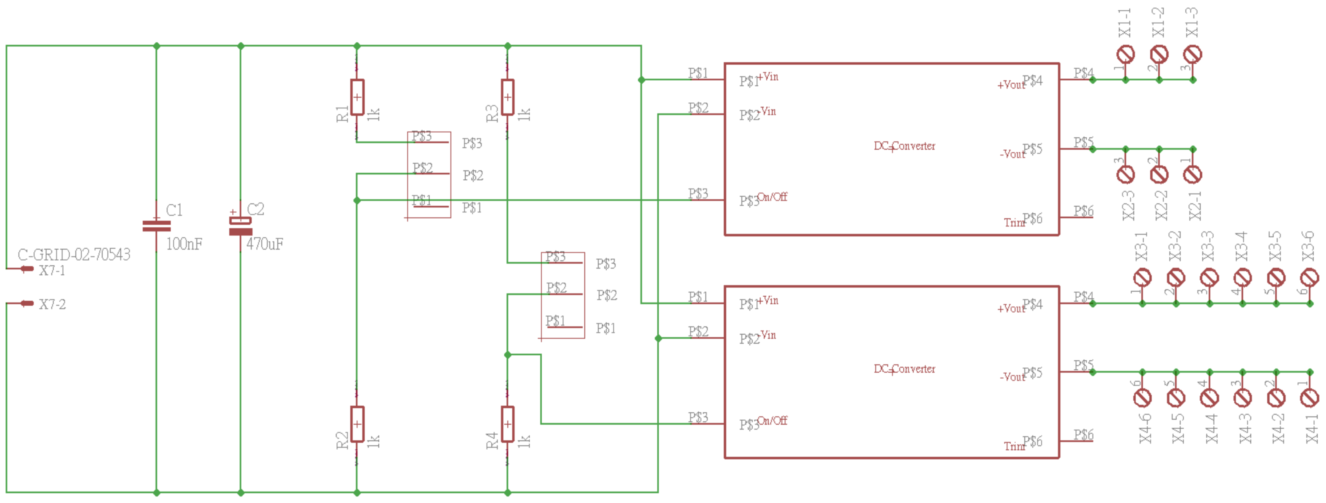


Figure 62 Power Distribution Circuit Schematic

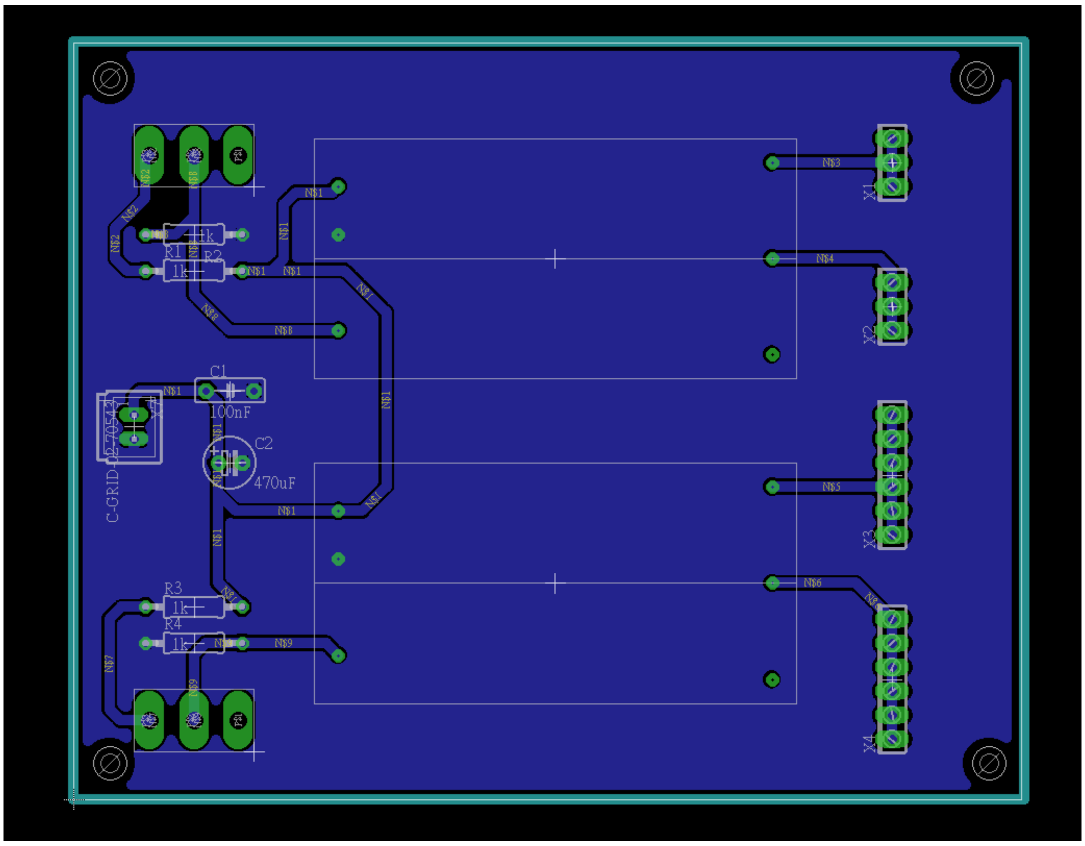


Figure 63 Power Distribution Circuit PCB Layout

Appendix G. Battery Monitoring

d. Hardware-based Schematic and PCB Design

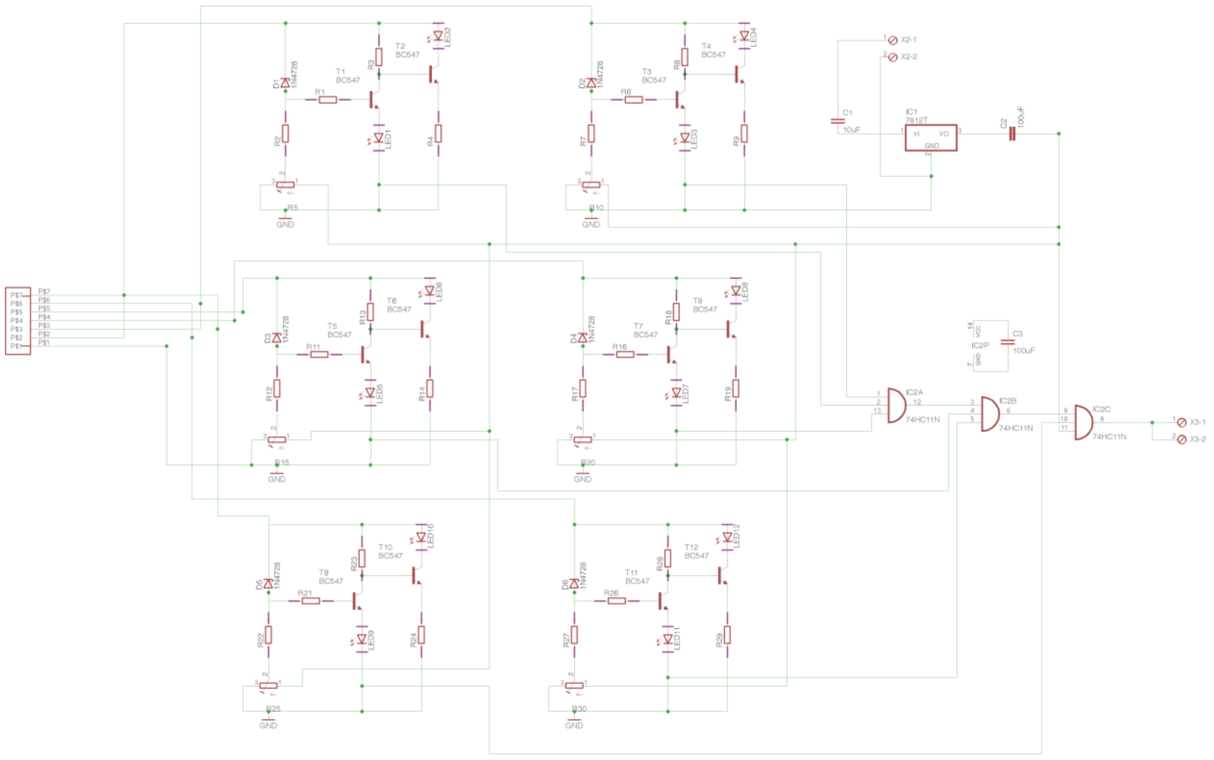


Figure 64 Hardware-based Battery Monitoring Circuit Schematic

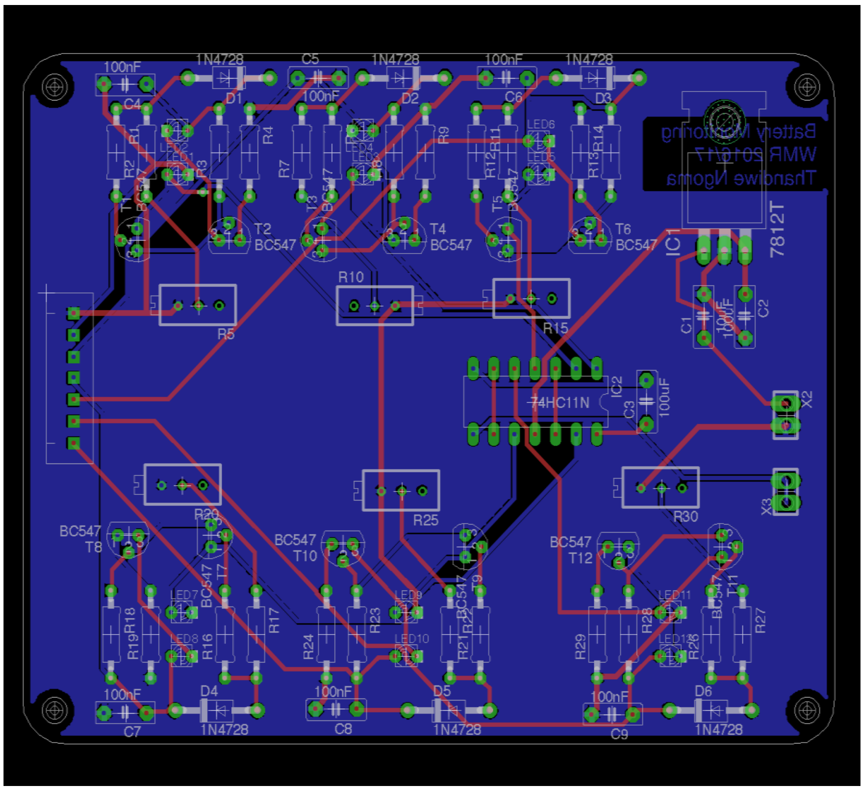


Figure 65 Hardware-based Battery Monitoring Circuit PCB Layout

e. Software-based Schematic and PCB

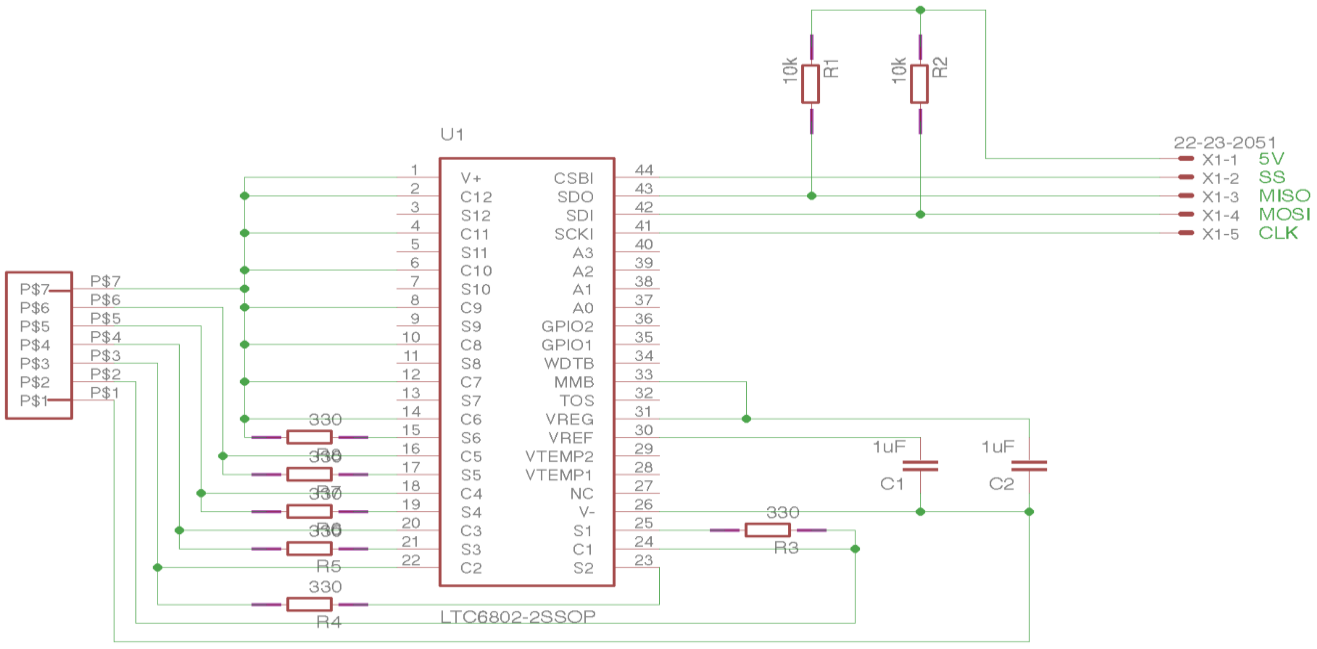


Figure 66 Software-based Battery Monitoring Circuit Schematic

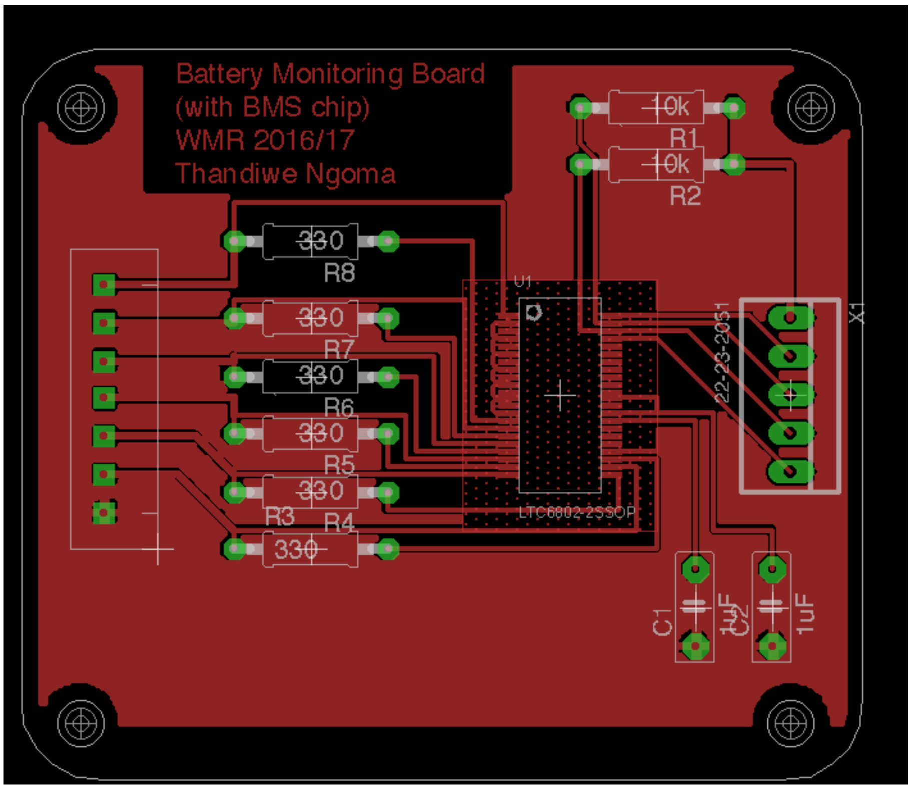


Figure 67 Software-based Battery Monitoring Circuit PCB Layout

Appendix H. Overcurrent Protection Circuit Schematic and PCB

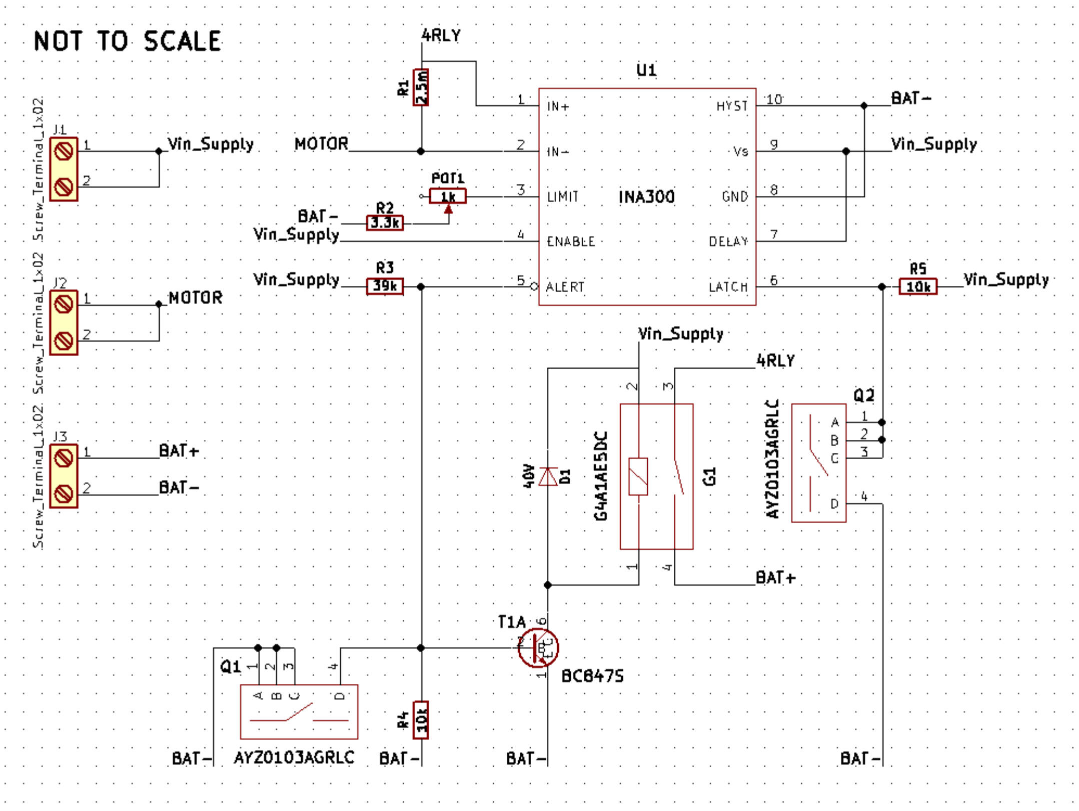


Figure 68 Overcurrent Protection Circuit Schematic

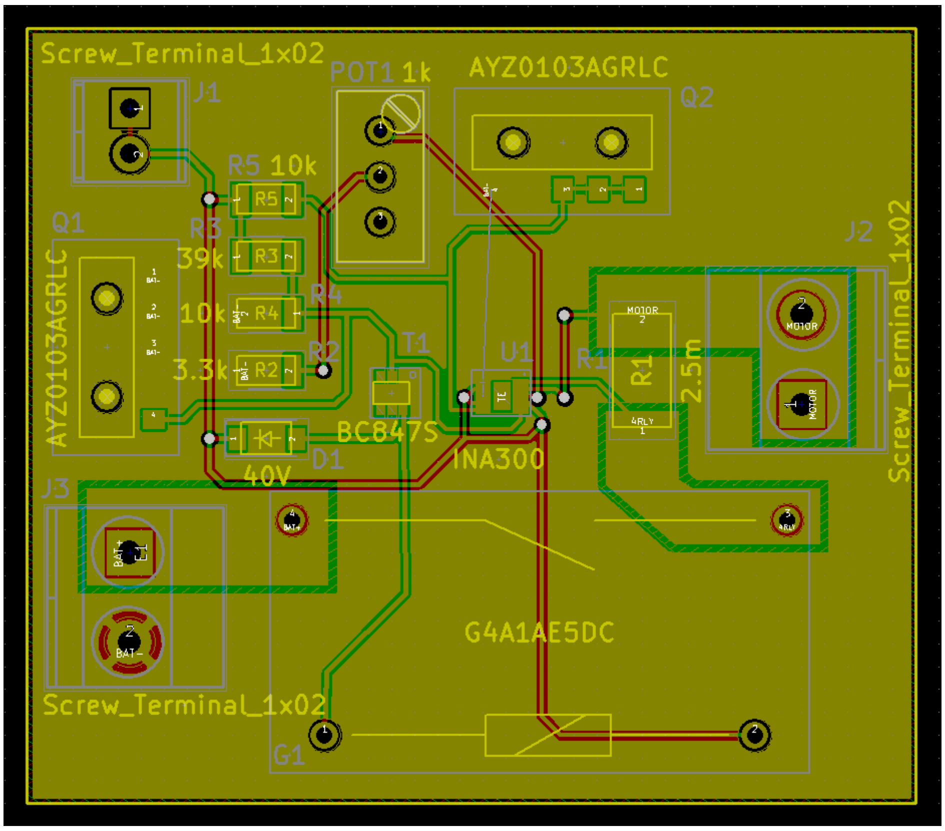


Figure 69 Overcurrent Protection Circuit PCB Layout

Appendix I. Motor Control

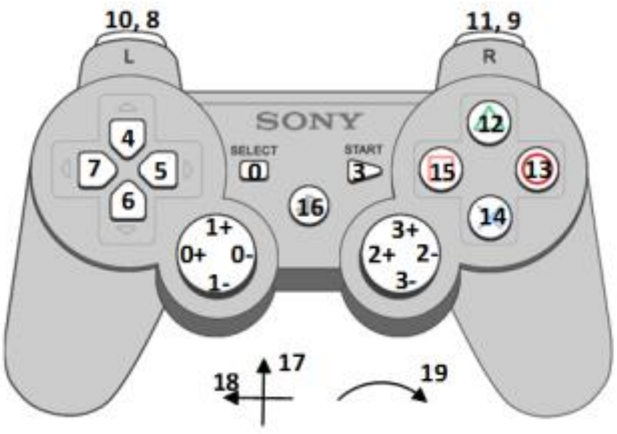


Figure 70 PS3 Controller's ID Reference Numbers

a. 'Move.cpp' C++ Script

```
//#include <ros/ros.h>
#include </opt/ros/indigo/include/ros/ros.h>
//#include <geometry_msgs/Twist.h>
#include </opt/ros/indigo/include/geometry_msgs/Twist.h>
//#include <sensor_msgs/Joy.h>
#include </opt/ros/indigo/include/sensor_msgs/Joy.h>


class MainTracks
{
public:
  MainTracks();

private:
  void joyCallback(const sensor_msgs::Joy::ConstPtr& joy);

  ros::NodeHandle nh_;

  int track_left, track_right, flipper_front_ac, flipper_front_c, flipper_rear_ac, flipper_rear_c;
  double l_scale_, a_scale_;
  ros::Publisher vel_pub_;
  ros::Subscriber joy_sub_;

//  int light;
//  ros::Publisher light_pub;

};


MainTracks::MainTracks():
  track_left(1),   // y-axis left analogue stick
  track_right(3),          // y-axis right analogue stick
  flipper_front_ac(8),     // front flipper anti-clockwise rotation. axis = 14, button = 8
  flipper_front_c(10),     // front flipper clockwise rotation. axis = 16, button = 10
  flipper_rear_ac(9),       //rear flipper anti-clockwise rotation. axis = 15, button = 9
  flipper_rear_c(11)        //rear flipper clockwise rotation. axis = 17, button = 11
 // light(12)

{

  nh_.param("axis_tl", track_left, track_left);
  nh_.param("axis_tr", track_right, track_right);
  nh_.param("button_ffa", flipper_front_ac, flipper_front_ac);
  nh_.param("button_ffc", flipper_front_c, flipper_front_c);
  nh_.param("button_fra", flipper_rear_ac, flipper_rear_ac);
  nh_.param("button_frc", flipper_rear_c, flipper_rear_c);
  nh_.param("scale_angular", a_scale_, a_scale_);
  nh_.param("scale_linear", l_scale_, l_scale_);

  vel_pub_ = nh_.advertise<geometry_msgs::Twist>("twist", 1);

  joy_sub_ = nh_.subscribe<sensor_msgs::Joy>("joy", 10, &MainTracks::joyCallback, this);
}

void MainTracks::joyCallback(const sensor_msgs::Joy::ConstPtr& joy)
{
    geometry_msgs::Twist twist;

    // Complex to put scaling here (msgs sent/time) - do on ardunio
    a_scale_ = 1;
    l_scale_ = 1;

    twist.linear.x = a_scale_*joy->axes[track_left];
    twist.linear.y = a_scale_*joy->axes[track_right];


    twist.angular.x = joy->buttons[flipper_front_ac];
    twist.angular.y = joy->buttons[flipper_front_c];
    twist.angular.z = joy->buttons[flipper_rear_ac];
    twist.linear.z = joy->buttons[flipper_rear_c];

    //publish the stated variables to topic 'twist'
    vel_pub_.publish(twist);

  //  std_msgs::UInt8 light;

  //  light.data = joy->buttons[12];
  //  light_pub.publish(light);

}


int main(int argc, char** argv)
{
  ros::init(argc, argv, "drive");
  MainTracks drive;


  ros::spin();
}
```

b. ROS Launch File - ‘move.launch’

```
<launch>

 <!-- Joy Node -->
  <node respawn="true" pkg="joy"
        type="joy_node" name="joystick">
  <param name="dev" type="string" value="/dev/input/js1" />
  <param name="deadzone" value="0.12" />
 </node>

 <!-- Axes -->
  <param name="axis_tl" value="1" type="int"/>
  <param name="axis_tr" value="3" type="int"/>
  <param name="button_ffa" value="8" type="int"/>
  <param name="button_ffc" value="10" type="int"/>
  <param name="button_fra" value="9" type="int"/>
  <param name="button_frc" value="11" type="int"/>
  <param name="scale_linear" value="2" type="double"/>
  <param name="scale_angular" value="2" type="double"/>

  <node pkg="atlas_drive" type="Move" name="drive"/>

 <!-- Ardunio Serial -->
  <node pkg="rosserial_python" type="serial_node.py" name="serial_node">
  <param name="port" value="/dev/ttyACM0"/>
  <param name="baud" value="57600"/>
 </node>

</launch>
```

c. Arduino Script – 'RoboClaw Compatible Motor Control.ido"

```
/* ------------------------------------------------------------ /
/                   Motor Control using PWM                   /
/                   ----------------------                   /
/ Motors included:                                            /
/  - Left Main Track Motor                                    /
/  - Right Main Track Motor                                   /
/  - Front Flipper Motor                                      /
/  - Rear Flipper Motor                                       /
/                                                             /
/ Subscribes to Topic "twist"                                 /
/ Written by: Thandiwe Ngoma                                  /
/-------------------------------------------------------------*/

#include <ros.h>
#include <std_msgs/UInt8.h>
#include <sensor_msgs/Joy.h>
#include <geometry_msgs/Twist.h>
#include <geometry_msgs/Vector3.h>
#include <std_msgs/UInt32.h>

#include <SoftwareSerial.h>
#include "RoboClaw.h"
//#include <BMSerial.h>

std_msgs::UInt32 Check;
std_msgs::UInt32 Check2;
ros::Publisher pub_check("OutputCheck", &Check);
ros::Publisher pub_check2("OutputCheck2", &Check2);

// limits at 10% and 90% of the threshold to ensure motors are not overused
#define UPPER_LIMIT 230
#define LOWER_LIMIT 25
```

```
//Address of Main Track's RoboClaw - 128 | mode 7
#define address 0x80
//Address of Flipper Arm's RoboClaw - 129 | mode 8
#define address2 0x81

/* ----------------------------------------------------------------
/    ROBOCLAW FUNCTIONS:
/ -> RoboClaw(uint8_t receivePin, uint8_t transmitPin, uint32_t tout)
/ -> bool ForwardM#(uint8_t address, uint8_t speed)
/ -> bool BackwardM#(uint8_t address, uint8_t speed)
/
/ Connect receivePin to S2 and transmitPin to S1
/ -----------------------------------------------------------------*/

SoftwareSerial serial(10, 11);
SoftwareSerial serial2(17, 16);
RoboClaw roboclaw(&serial,50);
RoboClaw roboclaw2(&serial2, 50);



//set variables to be used for main track code to hold signal values
int mod_value = 0;
unsigned int motor1_value = 0;
unsigned int motor2_value = 0;

//variables to check state of buttons controlling the flippers
int last_state1 = 0;
int last_state2 = 0;
int last_state3 = 0;
int last_state4 = 0;
int current_state1 = 0;
int current_state2 = 0;
int current_state3 = 0;
int current_state4 = 0;

//logic variables to set if there has been a change in button state
boolean pubf1 = true;
boolean pubf2 = true;
boolean pubr1 = true;
boolean pubr2 = true;

//set time variables to read millis() value.
//must be unsigned long to accommodate size of value the longer the program runs
unsigned long time1 = 0;
unsigned long time2 = 0;
unsigned long time3 = 0;
unsigned long time4 = 0;

// set a sensible amount of time to check for fluctuation in buttons pushed
unsigned long check = 50;

ros::NodeHandle  nh;

/* ---------------------------------------------------------------------------------/
/ RATIONALISE FUNCTION
/
/ Function to ensure the PWM value is within set limits to prevent overheating the motors
/ Taken from Daniel Riley's code (WMR 15/16)
/---------------------------------------------------------------------------------*/
int limits(int raw_value) {
  if (raw_value < LOWER_LIMIT){
    mod_value = LOWER_LIMIT;
  }
  else if (raw_value > UPPER_LIMIT) {
    mod_value = UPPER_LIMIT;
  }
  else {
    mod_value = raw_value;
  }
  return mod_value;
}

/*-----------------------------------------------------------------------------------------/
/ CALLBACK FUNCTION
/
/ Callback function to:
/ - read messages from topics
/ - scale joystick values (analogue -1:1 to 255:0)
/ - write PWM values to pins connected to RoboClaw controllers
/ Note:
/ - reverse values ~ as scale is 0-255, when joystick is pushed upwards, the values decrease
from 127 to 0
/   and increase from 127 to 255 when joystick is pushed downwards
/ - button state is either 0 (resting) or 1 (pressed)
/-----------------------------------------------------------------------------------------*/
void messageCb( const geometry_msgs::Twist& msg){

// ------ Begin Main Track Code ----------

  //Read message values for main track motors and store in float
  float m1 = msg.linear.x;
  float m2 = msg.linear.y;

  //scale the values -1:1 to 0 to 255
  //[-1 becomes 255 and 1 becomes 0]
  int m1_value_raw = ((m1*-127)+127);
  int m2_value_raw = ((m2*-127)+127);
  //run inputted value through limits function
  //limits(m1_value_raw);
  motor1_value = limits(m1_value_raw);
  //limits(m2_value_raw);
  motor2_value = limits(m2_value_raw);

//Write to Motor 1
if(motor1_value == 127 ) {
    roboclaw.ForwardM1(address, 0);
    Check.data = 0;
    pub_check.publish(&Check);
}
else {
    if(motor1_value < 127) {
      // Reverse values. As scale is 0-255, when joystick is pushed upwards, the values
decrease from 127 to 0.
      // Joystick value is subtracted from 127 so that speed is highest at the 'lowest' value.
      // Scaled for motor controllers that require 0-127 where 0 is STOP and 127 is FULL
FORWARD.
    int s = (127 - motor1_value);
    roboclaw.ForwardM1(address, s);
    Check.data = s;
    pub_check.publish(&Check);
    }
  else {
  //To put values in 0 - 127 scale, backward values are 128-254. Subtracting 127 from value
puts it in 0 - 127 scale
  // 0 is STOP and 127 is FULL REVERSE.
  int s = (motor1_value - 127);
    roboclaw.BackwardM1(address, s);
    Check.data = s;
    pub_check.publish(&Check);
   }
}

//Write to Motor 2
if(motor2_value == 127) {
  roboclaw.BackwardM2(address, 0);
  Check2.data = 0;
    pub_check2.publish(&Check2);
}
else {
    if(motor2_value < 127) {
      // Reverse values. As scale is 0-255, when joystick is pushed upwards, the values
decrease from 127 to 0.
      // Joystick value is subtracted from 255 so that speed is highest at the 'lowest' value.
      // Scaled for motor controllers that require 0-127 where 0 is STOP and 127 is FULL
FORWARD.
      roboclaw.BackwardM2(address, (127 - motor2_value));
      Check2.data = (127 - motor2_value);
      pub_check2.publish(&Check2);
    }
   else {
      //To put values in 0 - 127 scale, backward values are 128-254. Subtracting 127 from
value puts it in 0 - 127 scale
      // 0 is STOP and 127 is FULL REVERSE.
      roboclaw.ForwardM2(address, (motor2_value - 127));
       Check2.data = (motor2_value - 127);
       pub_check2.publish(&Check2);
   }
  }

// ------ Begin Flipper Code ----------

  //Read message values for the flipper motors and store in floats
  float ff1_raw = msg.angular.x;
  float ff2_raw = msg.angular.y;
  float rf1_raw = msg.angular.z;
  float rf2_raw = msg.linear.z;



    //Set initial state of each button
    int current_state1 = ff1_raw;
    int current_state2 = ff2_raw;
    int current_state3 = rf1_raw;
    int current_state4 = rf2_raw;

// LOOP 1 - Front Flipper Forward Rotation
    //check to see if there has been a change in button state
    if (last_state1 != current_state1) {
      time1 = millis();
      pubf1 = false;
    }

    //If the change in signal is longer than the debounce limit set to allow the push button to
settle, change the state of the flipper output pin
    if(!pubf1 && ((millis() - time1) > check)) {
      if (current_state1 > 0.5) {
        //Front Flipper Forward Rotation
        roboclaw2.ForwardM1(address2, 100);
      }
      else {
      //Front Flipper Stop
        roboclaw2.ForwardM1(address2, 0);
      }
      pubf1 = true;
    }
    last_state1 = current_state1;
```

```
//LOOP 2 - Front Flipper Backwards Rotation
    //check to see if there has been a change in button state
    if (last_state2 != current_state2) {
      time2 = millis();
      pubf2 = false;
    }

    //If the change in signal is longer than the debounce limit set to allow the push button to
settle, change the state of the flipper output pin
    if(!pubf2 && ((millis() - time2) > check)) {
      if (current_state2 > 0.5) {
        //Front Flipper Backward Rotation
        roboclaw2.BackwardM1(address2, 100);
      }
      else {
        //Front Flipper Stop
        roboclaw2.BackwardM1(address2, 0);
      }
      pubf2 = true;
    }
    last_state2 = current_state2;

// LOOP 3 - Rear Flipper Forward Rotation
    //check to see if there has been a change in button state
    if (last_state3 != current_state3) {
      time3 = millis();
      pubr1 = false;
    }

    //If the change in signal is longer than the debounce limit set to allow the push button to
settle, change the state of the flipper output pin
    if(!pubr1 && ((millis() - time3) > check)) {
      if (current_state3 > 0.5) {
        //Rear Flipper Forward Rotation
        roboclaw2.ForwardM2(address2, 64);
      }
      else {
        //Rear Flipper Stop
        roboclaw2.ForwardM2(address2, 0);
      }
      pubr1 = true;
    }
    last_state3 = current_state3;

// LOOP 4 - Rear Flipper Backwards Rotation
    //check to see if there has been a change in button state
    if (last_state4 != current_state4) {
      time4 = millis();
      pubr2 = false;
    }

    //If the change in signal is longer than the debounce limit set to allow the push button to
settle, change the state of the flipper output pin
    if(!pubr2 && ((millis() - time4) > check)) {
      if (current_state4 > 0.5) {
        //Rear Flipper Backward Rotation
        roboclaw2.BackwardM2(address2, 64);
      }
      else {
        //Rear Flipper Stop
        roboclaw2.BackwardM2(address2, 0);
      }
      pubr2 = true;
    }
    last_state4 = current_state4;
```

```
}

//Subscribe to ROS topic "twist"
ros::Subscriber<geometry_msgs::Twist> sub("twist", &messageCb);

//Setup code that only runs once
void setup() {
  //Initialise nodes and subscribe to topics
  nh.initNode();
  nh.subscribe(sub);
  nh.advertise(pub_check);
  nh.advertise(pub_check2);

  //Open roboclaw serial ports
  roboclaw.begin(9600);
  roboclaw2.begin(9600);
}

//Main code to run repeatedly
void loop() {
  nh.spinOnce();
  delay(1);
}
```

Appendix J. CO$_2$ Sensor Code

```
/* ------------------------------------------------------------------------------------------------------
-------------
 *   CO2 SENSING
 * ------------------------------------------------------------------------------------------------------
-------------
```

```
 *  Written by Thandiwe Ngoma
 * Reference ->
 *  -
 *  Title: Demo for MG-811 Gas Sensor Module V1.1
 *  Author:Tiequan Shao: tiequan.shao@sandboxelectronics.com
          Peng Wei:     peng.wei@sandboxelectronics.com
 *  Date: 2012-05-31
 *  Availability:https://www.dfrobot.com/wiki/index.php/CO2_Sensor_SKU:SEN0159
 *  -
 * --------------------------------------------------------------------------------------------------------
------------*/

/* --------------------------------------------------------------
 * ROS Setup
 * --------------------------------------------------------------
 * Node handle is defined - nh
 * std_msgs/Uint32 library to be used to send voltage and concentration percentage values
 * Publisher will be pub_CO2 and will publish to topic 'CO2value'
 * --------------------------------------------------------------*/
#include <ros.h>
#include <std_msgs/UInt32.h>

ros::NodeHandle  nh;

std_msgs::UInt32 CO2_value;
ros::Publisher pub_CO2("CO2value", &CO2_value);

/*----------------------------------------------------------------
 * PIN CONNECTIONS AND VARIABLE SETUP
 *----------------------------------------------------------------*/
//Define the pins to be connected
//Analogue input pin
#define A_in  0
```

```
//Digital pin to read sensors digital ouput - this is a boolean pin so will either be HIGH or
LOW
#define D_in 53
//Amplifier gain value
#define DC_GAIN 8.5

//Define the sample limits
//Number of samples to be taken
#define SAMPLE_NUMBER 10
//Time interval between each sample in milliseconds
#define SAMPLE_INTERVAL 50

//set variables to hold analogue voltage values and concentration percentage values
int percentage;
float volts;

/* --------------------------------------------------------
 * Setting up the CO2 Curve
 * --------------------------------------------------------
 * CO2 Curve format:{ x, y, slope};
 * First point x: (lg400=2.602, MIN_CONC_VOLTAGE)
 * Second point y: (lg10000=4, MAX_CONC_VOLTAGE)
 * Slope: (y1-y2)/(x1-x2)
 * n.b. (y1-y2) is equal to the reaction voltage (voltage drop from moving the sensor from
air in to 10000ppm
 *      (x1-x2) is the operating range of the sensor
 *
 * More precise readings can be taken if more points on the curve are plotted.
 * --------------------------------------------------------*/
//Define the values to set up the CO2 Curve
//The start point_on X_axis of the curve
#define ZERO_POINT_X  2.602
//The output of the sensor in volts when the concentration of CO2 is 400PPM
```

```
//VALUE TO CHANGE WITH CALIBRATION
#define MIN_CONC_VOLTAGE 0.324
//The output of the sensor in volts when the concentration of CO2 is 10,000PPM
//VALUE TO CHANGE WITH CALIBRATION
#define MAX_CONC_VOLTAGE 0.265
//The voltage drop€ of the sensor when moved from air into 1000ppm CO2
// Equal to [MIN_CONC_VOLTAGE - MAX_CONC_VOLTAGE]
//VALUE TO CHANGE WITH CALIBRATION
#define REACTION_VOLTGAE 0.059

//Define x, y and the slope of the CO2 Curve
float        CO2Curve[3]  =  {ZERO_POINT_X, MIN_CONC_VOLTAGE,
(REACTION_VOLTGAE / (2.602 - 4))};

void setup() {

//Set input pin
pinMode(D_in, INPUT);
//Set digital pin to HIGH to start. Turns on pullup resistors
digitalWrite(D_in, HIGH);

//Initialise Node and advertise published topic
  nh.initNode();
  nh.advertise(pub_CO2);
}

/* -------------------------------------------------------------------
 * Main Loop
 * -------------------------------------------------------------------
 * Calls to each function and publishes the results in ROS
 * -------------------------------------------------------------------*/

void loop() {
 volts = SensorRead(A_in);
 //Publish voltage read to ROS
     CO2_value.data = volts;
     pub_CO2.publish(&CO2_value);

 percentage = FindPercentage(volts, CO2Curve)
 //Publish percentage gotten to ROS
     CO2_value.data = percentage;
     pub_CO2.publish(&CO2_value);

//Publish time point
     CO2_value.data = millis();
     pub_CO2.publish(&CO2_value);

}

/* ---------------------------------------------------------------------
 * SENSOR READ FUNCTION
 * ---------------------------------------------------------------------
 * This function reads the analogue output from the sensor, takes 10 samples and
 * finds the average
 *
 * Note: The analogue input is divided in to 1024 discrete output values by the ADC
 *       so the input is read in the range of 0 to 1023. The voltage requires dividing
 *       by 1024 to get the percentage that can be multiplied by 5 to give an output
 *       voltage in the Arduino's analogue input range of 0 to 5V.
 * --------------------------------------------------------------------*/

float SensorRead(int a_in ) {
  int i;
  float v = 0;

  for (i = 0; i < SAMPLE_NUMBER; i++) {
    v += analogRead(a_in);
    delay(SAMPLE_INTERVAL);
  }
  v = (v / SAMPLE_NUMBER) * 5 / 1024 ;
  return v;
}


/* ---------------------------------------------------------------------
 * FIND PERCENTAGE FUNCTION
 * --------------------------------------------------------------------
 * This function takes the sampled voltage float value and reads the exponent
 * off of the CO2 curve to find the corresponding concentration percentage
 *
 * pcurve is used as a pointer to the curve of the target gas
 * --------------------------------------------------------------------*/
int FindPercentage( float volts, float *pcurve) {
   volts = volts / DC_GAIN;
  if (volts > MIN_CONC_VOLTAGE || volts < MAX_CONC_VOLTAGE ) {
    return -1;
  } else {
    //power of 10 function is used to convert the result out to non-logarithmic values
    return pow(10, (volts - pcurve[1]) / pcurve[2] + pcurve[0]);
    //reset volts for next set of readings
    volts = 0;
  }
}
```

Appendix K. Battery Monitoring Code

```
/* ----------------------------------------------------------------------------------------------------
-------------
 *  BATTERY MONITORING CODE FOR LTC6802-2 CHIP
 * ----------------------------------------------------------------------------------------------------------
-------------
 *  Written by Thandiwe Ngoma
 *  Using example code taken from https://sourcelion.wordpress.com/2014/07/20/battery-
management-system-ltc6802-arduino/
 *
 *  NOTE: The chip can be used with as few as 4 cells but requires a minimum operating
voltage of 10V to guarantee electrical
 *       specifications are met.
 * ----------------------------------------------------------------------------------------------------------
------------*/

//Libraries to include
#include <SPI.h>

//Include the appropriate ros directories and initiate the node handle
#include <ros.h>
#include <std_msgs/Int32.h>

ros::NodeHandle nh;

std_msgs::Int32 voltage;
ros::Publisher pub_CV("cellV", &voltage);



/* ---------------------------------------------------------------------------------
 * Pre-set all LTC6802-2 Chip Registers. Abbreviations in keeping with the datasheet
 * --------------------------------------------------------------------------------
 * WRCFG - Write Configuration Register Group
 * RDCFG - Read Configuration Register Group
 * RDCV - Read Cell Voltage Register Group
 * RDFLG - Read Flag Register Group
 * STCVAD - Start Cell Voltage A/D Conversions and Poll Status
 * ----------------------------------------------------------------------------*/

#define WRCFG 0x01
#define RDCFG 0x02
#define RDCV 0x04
#define RDFLG 0x06
#define STCVAD 0x10

//Address is set to 0x80 as only one LTC chip is currently being used
#define address 0x80

int i = 0;
int j = 0;
int k = 0;

int CV1_value = 0;
int CV2_value = 0;
int CV3_value = 0;
int CV4_value = 0;
int CV5_value = 0;
int CV6_value = 0;

//To convert voltages correctly must be multiplied by 1.5mV
int convert = 1.5*0.001;

/* SPI pins for the Arduino Mega are:
  *  MOSI - 51 (this is SDI in the chip datasheet)
  *  MISO - 50 (this is SDO in the chip datasheet)
  *  SCK - 52  (thid id SCKI in the chip datasheet)
  *  SS - 53   (this is CSBI in the chip datasheet)
  *  Set SPI connector pins
  *  LTC6802-2 Chip (Slave) will only communicate with the Arduino(Master) when the
SS pin is LOW.
```

```
*/

int miso = 50;
int mosi = 51;
int sck = 52;
int ss = 53;

void setup() {

  pinMode(miso, OUTPUT);
  pinMode(mosi, OUTPUT);
  pinMode(SCK, OUTPUT);
  pinMode(ss, OUTPUT);

  //Set SS as HIGH to prevent communication from the start.
  digitalWrite(ss, HIGH);
  //Set SPI Parameters specific to LTC6802-2 chip
  // Clock Phase(CPHA) and Polarity(CPOL) both equal 1, so the transmission mode needs
to be set to 3
  SPI.setDataMode(SPI_MODE3);
  //The data transfers with the most significant bit (MSB) first.
  //note: every byte consists of 8 bits
  SPI.setBitOrder(MSBFIRST);
  SPI.setClockDivider(SPI_CLOCK_DIV16);
  Serial.begin(57600);
  SPI.begin();
  //Run function to write to registers
  writeReg();

  nh.initNode();
  nh.advertise(pub_CV);

}
```

```
void loop() {
  readVoltage();
  nh.spinOnce();
  delay(2000);
}

/* -----------------------------------------------------------------------------------------------
 *        WRITE CONFIGURATION REGISTER
 * -----------------------------------------------------------------------------------------------
 * Setup Registers with the values of the Over and Under voltage conditions
 * -----------------------------------------------------------------------------------------------*/
void writeReg()
{
 //START data transfer by setting Slave Select to LOW
 digitalWrite(ss, LOW);
 //Always give the Address of the LTC6802-2 chip first followed by the Command Address
 SPI.transfer(address);
 //Command is the WRITE command which writes the value of each configuration register
 SPI.transfer(WRCFG);

 /*Set all Cell registers to 0V, the Under Voltage (VUV) register to 3.3V and the Over
Voltage (VOV) to 4.2V
 *UNDERVOLTAGE: In the datasheet, to find the comparison voltage (3.3V in this case),
the following equation must be used to determine VUV
 *    3.3V = VUV x 16 x 1.5mV
 *    VUV = 137.5 (DECIMAL) so therefore VUV = 0x89 (HEX)
 *OVERVOLTAGE: In the datasheet, to find the comparison voltage (4.2V in this case),
the following equation must be used to determine VOV
 *    4.2V = VOV x 16 x 1.5mV
 *    VOV = 175 (DECIMAL) so therefore VOV = 0xAF (HEX)
 */
 //Register 0
 SPI.transfer(0x01);
 //Register 1
 SPI.transfer(0x00);
 //Register 2
 SPI.transfer(0x00);
 //Register 3
 SPI.transfer(0x00);
 //Register 4
 SPI.transfer(0x00);
 //Register 5
 SPI.transfer(0x00);
 //VUV Register - Register 6
 SPI.transfer(0x89);
 //VOV Register - Register 7
 SPI.transfer(0xAF);

 //END data transfer by setting Slave Select to HIGH
 digitalWrite(ss, HIGH);
}

/* ----------------------------------------------------------------------------------------------------
 *        READ CONFIGURATION REGISTER
 * ----------------------------------------------------------------------------------------------------
 * This function reads registers to ensure they have been properly configured in the
'writeReg()' function
 * ----------------------------------------------------------------------------------------------------*/
void readReg() {
  //Set variable to store data read from registers
  byte Check;
  //START data transfer by setting Slave Select to LOW
  digitalWrite(ss, LOW);
  //Always give the Address of the LTC6802-2 chip first followed by the Command
Address
  SPI.transfer(address);
  //Command is the READ command which reads the value of each configuration register
  SPI.transfer(RDCFG);
  //Loop through all set registers. 8 registers were set so loops 8 times
  //Should return 6 empty registers and the VUV and VOV values.
  //Remember - Values will be in HEXDECIMAL form
  for(int i = 0; i < 8; i++)
  {
  Check = SPI.transfer(RDCFG);
  Serial.println(Check, HEX);
  }

  //END data transfer by setting Slave Select to HIGH
  digitalWrite(ss, HIGH);
}

/* -----------------------------------------------------------------------------------------------------
-------------
 *       READ CELL VOLTAGE REGISTER
 * -----------------------------------------------------------------------------------------------------
-------------
 *  This function monitors the voltage of each cell by reading the values in the Read Cell
Voltage Configuration Registers
 *  LTC6802-2 has the capability to read up to 12 cell voltages stored over 18 registers
 *  However, as only 6 cells require monitoring whilst only one battery is used a variable
array is set to 8
 *  (Voltage value is spread over 12bits but each register is only 8bits so each voltage is
spread across 2 registers)
 *
 *  The code to add the 8 bits from one register with 4 bits from another was taken from the
previously mentioned blog post:
 *  https://sourcelion.wordpress.com/2014/07/20/battery-management-system-ltc6802-
arduino/
 * -----------------------------------------------------------------------------------------------------------------
------------*/
void readVoltage() {
  //START data transfer by setting Slave Select to LOW
  digitalWrite(ss, LOW);
  //Start the Analog to Digital conversion of the cell voltages - ALL devices should start
converting simultaneously
  //Required to send this command to intiate cell voltage measurements.
  SPI.transfer(STCVAD);
  //MISO output is temporarily pulled low for ~12ms so a delay is required
  //Delay of 16 ms set so as to give 25% leeway given that 12ms is only an approximation
but do not want to cause further delays if neccessary
  delay(16);
  //TEMPORARY END to data transfer by setting Slave Select to HIGH
  digitalWrite(ss, HIGH);

  //Set variable with an array of 8 to store data read from Cell Voltage (CV) registers
  byte volt[8];
  //START data transfer by setting Slave Select to LOW
  digitalWrite(ss, LOW);
  //Always give the Address of the LTC6802-2 chip first followed by the Command
Address
  SPI.transfer(address);
  //Command is the READ command which reads the value of each CELL VOLTAGE
configuration register
  SPI.transfer(RDCV);
  //Start a loop to store the CV values in each part of the array
  for (j = 0; j<8; j++)
  {
    volt[j] = SPI.transfer(RDCV);
  }
```

```
   //END data transfer by setting Slave Select to HIGH
   digitalWrite(ss, HIGH);

   //To read the correct voltage, values must be multiplied by 1.5mV and the full 12 bit
voltage from multiple registers is brought together
  CV1_value = ((volt[0] & 0xFF) | (volt[1] & 0x0F) << 8) * convert;
  CV2_value = ((volt[1] & 0xF0) >> 4 | (volt[2] & 0xFF) << 4) * convert;
  CV3_value = ((volt[3] & 0xFF) | (volt[4]  & 0x0F) << 8) * convert;
  CV4_value = ((volt[4] & 0xF0) >> 4 | (volt[5] & 0xFF) << 4) * convert;
  CV5_value = ((volt[6] & 0xFF) | (volt[7] & 0x0F) << 4) * convert;
  CV6_value = ((volt[7] & 0xF0) >> 4 | (volt[8] & 0xFF) << 4) * convert;

  //Print values on ROS topic 'cellV'
  voltage.data = CV1_value;
  voltage.data = CV2_value;
  voltage.data = CV3_value;
  voltage.data = CV4_value;
  voltage.data = CV5_value;
  voltage.data = CV6_value;

  pub_CV.publish(&voltage);

  //Print values when not running in ROS
  Serial.println(CV1_value);
  Serial.println(CV2_value);
  Serial.println(CV3_value);
  Serial.println(CV4_value);
  Serial.println(CV5_value);
  Serial.println(CV6_value);
  Serial.println("-------------------");
}

/* -----------------------------------------------------------------------------------------------------
```

```
 *      READ FLAG REGISTER
 * -----------------------------------------------------------------------------------------------------
 * This function checks that no FLAGS have gone off to indicate one of the cells is over or
under voltage.
 * If the a cell has been flagged then a red LED is set off
 * -----------------------------------------------------------------------------------------------------*/

void checkFlag()
{
    //set variable
    byte flag[2];
   //START data transfer by setting Slave Select to LOW
   digitalWrite(ss, LOW);

   //Always give the Address of the LTC6802-2 chip first followed by the Command
Address
   SPI.transfer(address);
   //Command is the READ command which reads the Flag register
   SPI.transfer(RDFLG);
   for (k = 0; k <2; k++ )
   {
    flag[k] = SPI.transfer(RDFLG);
   }
   //END data transfer by setting Slave Select to HIGH
   digitalWrite(ss, HIGH);
}
```