UK CanSat Competition 2013-14

Final report

TEAM NOVA

Abstract

This report discusses Team Nova's entry to the UK CanSat Competition 2013-14.

A CanSat Competition is a competition in which teams compete to construct an experiment the size of a drinks can (a 'CanSat'), which is then launched to an altitude of up to one kilometre. The CanSat must record temperature and pressure data and send the data to a base station every second (the 'Primary Mission'), and must also conduct another mission of the team's choosing (the 'Secondary Mission').

Team Nova's selected secondary mission was to construct a multi-terrain rover capable of two-way telecommunications; this could potentially have applications in space exploration, military reconnaissance and the monitoring/exploration of remote regions.

The rover uses a tracked design with a simple drivetrain to maximise space available for sensors, processing and communications equipment. The report details the design process involved in the creation of the mechanical/structural parts of the CanSat and the electronic parts of the CanSat. The descent system (involving a parachute and a landing module) is also discussed.

Throughout the project, outreach was conducted both through online and offline mediums; this aimed to raise the profile of the UK CanSat Competition and engineering in general. The report records the key channels through which outreach was conducted. Project management (including planning, resource management and testing procedures) and the events surrounding the launch of the CanSat are also covered. Finally, the report contains reflections on all parts of the project (including fulfilment of objectives, project management and outreach).

Contents

1 Introduction and Preamble

1.1 Description of the CanSat Competition

CanSat competitions involve the production of a small payload the size of a can, which is then launched to an altitude of up to a kilometre. The UK CanSat Competition requires each CanSat to conduct a set primary mission and select a secondary mission.

The primary mission was to record temperature and pressure sensor during the descent, and to report the measurements over radio to a 'base station' at a frequency of 1Hz. Our chosen secondary mission was to construct a rover capable of two way communication with the base station.

We entered in the advanced category, which has the same requirements but indicated that we had had some prior experience.

1.2 Team Members and Roles

Our team comprised of 7 students. We organised ourselves into three subgroups which operated semi-autonomously, as shown below. This improved the efficiency with which we could carry out tasks, as it was easier to trace responsibility. Some team members had additional responsibilities as indicated.

Theo Heymann - team manager

Electronics, programming and data processing
David Shah - group leader
Jameson Lee - responsible for base station software design

Structural and mechanical design, and manufacture
Nicolas Weninger - group leader
Mark Bobrovnikov - responsible for ground support station mechanical design
Matin Alimadadian - responsible for parachute design

Outreach
Josh Efiong - group leader

1.3 Mission Outlines

1.3.1 Primary Mission Objective

Record temperature and pressure data and transmit that data to the base station every second.

1.3.2 Secondary Mission Objective

Deploy a semi-autonomous rover capable of two-way communication with a base station.

1.3.3 Criteria for Success in Secondary Mission

There were six main technical challenges that we intended to investigate and surmount. Fulfilment of these would mean the project had been successful. The criteria were as follows, and are stated in their original wording.

1. Designing the vehicle to be sufficiently compact that it fits within the size and weight limitations of the CanSat. This will involve the very careful choice and design of electronic and mechanical parts to reduce weight, size and power consumption.
2. Reducing damage during landing so the vehicle is able to function normally after a standard landing procedure.
3. Traversal of a wide variety of terrain, including short grass and a variety of manmade surfaces.
4. Two way communication with the base station with minimal latency, including a video stream and sensor readings from the CanSat to the base station, and commands from the base station to the CanSat.
5. Both semi-autonomous and manual control of the rover's movement. If the rover loses connection, it will initially attempt to regain connection by driving towards the base station.
6. Developing an interface for use on the base station. This will feature a video feed from the onboard camera, the ability to control the vehicle and a feed of information from the sensors, as well as a satellite or map view of the CanSat's location.

1.3.4 Reasons for Selecting this Secondary Mission

Miniature rovers have numerous potential applications.

- Transporting a small rover on an extra-terrestrial probe would require much less energy than transporting a large probe, and could also be used as a separate, or even disposable, device for moving into small spaces. A rover of that variety would likely only require short-range telecommunications, as data would be relayed through another vehicle or probe before being sent back to earth.
- Military reconnaissance missions could potentially be aided by the use of miniature rovers; these are able to move around without attracting much notice. Such rovers could be dropped by military aircraft and record and transmit video and audio data.
- Exploration and research teams could also make use of this type of vehicle. For example, they could be used to provide on-the-ground images and data in remote locations.

1.4 Fundamental Specifications

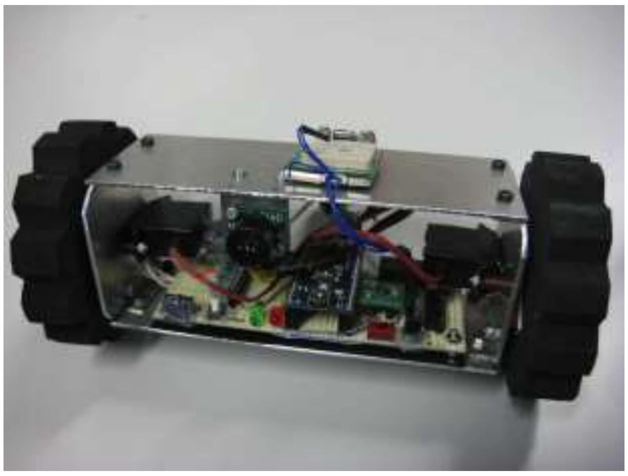
The specification as provided by the organisers of the UK CanSat Competition is below:

1. All the components of the CanSat must fit inside a standard soda can (115 mm height and 66 mm diameter), with the exception of the parachute. An exemption can be made for radio antennas and GPS antennas, which can be mounted externally (on the top or bottom of the can, not on the sides), based on the design. N.B. In the case of the T-Minus Engineering rockets, the space allocated for each CanSat is 66 mm diameter and a height of 200 mm. The extra height can ONLY be used for the exemptions mentioned.

2. The antennas, transducers and other elements of the CanSat cannot extend beyond the can's diameter until it has left the launch vehicle.
3. The mass of the CanSat must be 370 g. CanSats that are lighter must take additional ballast with them to reach the 370g mass limit required.
4. Explosives, detonators, pyrotechnics, and flammable or dangerous materials are strictly forbidden. All materials used must be safe for the personnel, the equipment and the environment. Material Safety Data Sheets (MSDS) will be requested in case of doubt.
5. The CanSat must be powered by a battery and/or solar panels. It must be possible for the systems to be switched on for three continuous hours.
6. The battery must be easily accessible, in case it has to be replaced or recharged in the field.
7. The CanSat must have an easily accessible master power switch.
8. The CanSat should have a recovery system, such as a parachute, which is able to be reused after launch. It is recommended to use bright coloured fabric, which will facilitate recovery of the CanSat after landing.
9. The parachute connection must be able to withstand up to 1000N of force. The strength of the parachute must be tested, to give confidence that the system will operate nominally.
10. The decent time of the CanSat when falling from 1000 meters is limited to 90 seconds. This is a requirement from the launch site since this will guaranty that the CanSat lands within the landing area under all wind conditions.
11. The descent rate must be at least 11m/s.
12. The CanSat must be able to withstand an acceleration of up to 20g.
13. The recovery of the CanSat is not guaranteed after the launch.
14. The total budget of the CanSat should not exceed £400 (Beginners' category) or £500 (Advanced category).

1.5 Initial Research

The best documented rover CanSat we were able to find was that created by an international team on the CanSat Leadership Training Program in Wakayama University.[1]



The rover design (above) eventually selected was to place wheels on each side of the CanSat, with an aluminium body between them. This body contained all the electronics, which included a standalone

[1] See: http://cltp.info/pdf/paper3.pdf [accessed 02/09/2013]

camera, a cutter for removing the parachute upon landing, a GPS module and an XBee 2.4Ghz radio module.

The team reported in a paper that their CanSat was able to carry out instructions communicated remotely to it.

Analysis of the design would, however, suggest that it would have poor performance over rough terrain and in places with a significant amount of water on the ground, on account of the low chassis height and exposed nature of the electronics.

We also found a number of similar designs, with two wheels on the ends of the CanSat (see https://www.youtube.com/watch?v=DXSd4BaqBF4, https://www.youtube.com/watch?v=rL_EOPnpDWY and https://www.youtube.com/watch?v=W-lh4i50YzY); we were unable to find any designs operating on a distinctly different principle.

2 CanSat Development

2.1 Final Project Overview

2.1.1 Final CanSat Overview

The CanSat itself features two main units: the rover itself (the focus of our secondary mission) and the landing module. The rover descends from the launch altitude within the landing module; upon landing, the landing module opens and releases the rover, which can then move around semi-autonomously and stream data over Wi-Fi. The design of the rover itself is discussed in *section 2.2* and *2.3*; the landing module is discussed in *section 2.5*.

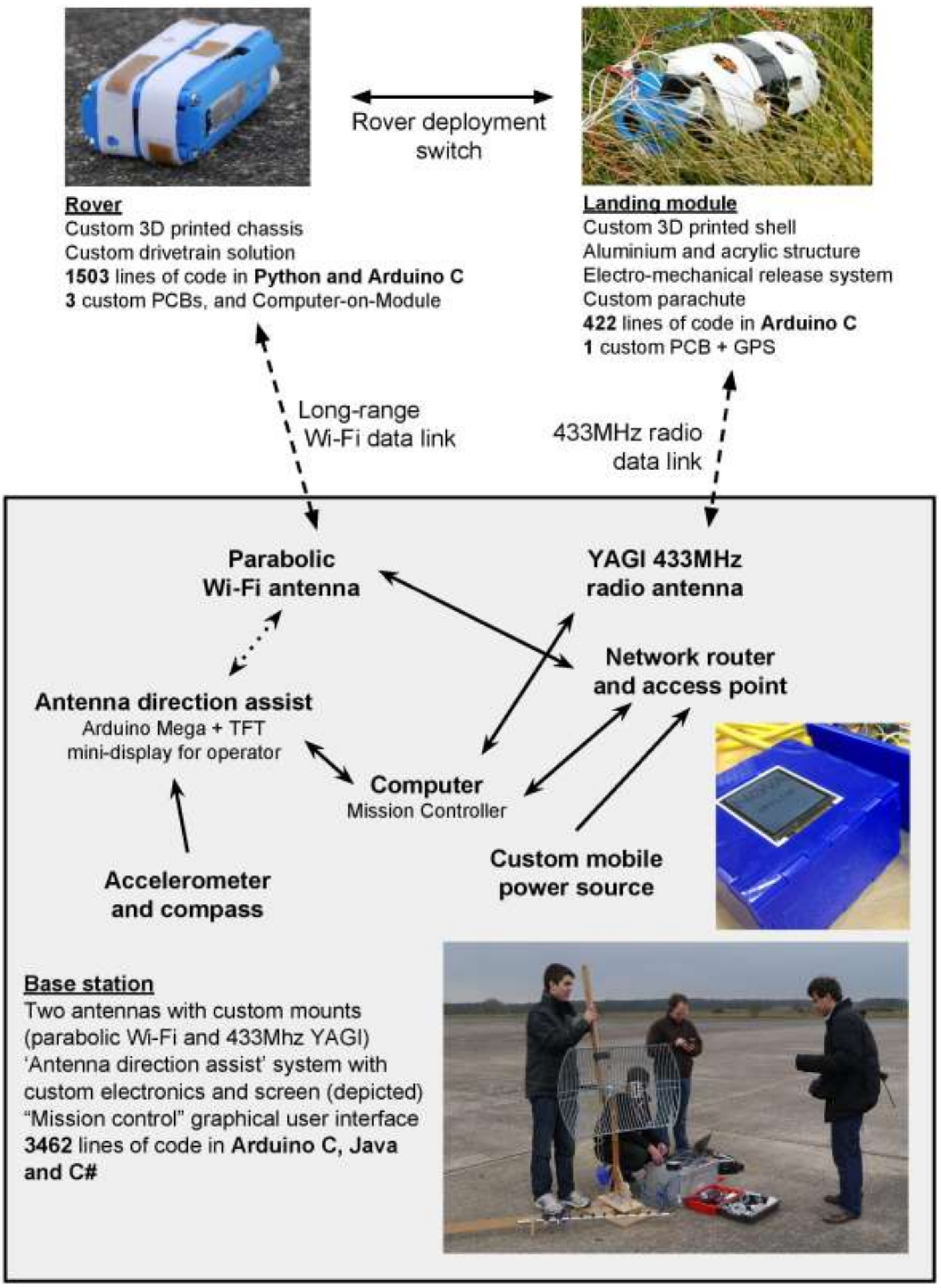
Because the aim of the CanSat is based around the rover, we consider the landing module as part of the landing and recovery system, rather than as a part of the CanSat itself.

2.1.2 Base Station Overview

Mission control is conducted using custom software installed on a laptop as part of the 'base station'. The laptop is connected to several other components, most notably a parabolic WiFi antenna (via a network router), an electronic 'direction assisst' system (to help with pointing the WiFi antenna) and a radio module transmitting and receiving using a YAGI antenna. The base station is discussed further in *section 2.6*.

2.1.3 Hardware Summary

The following schematic provides an overview of all parts of the project and how they interlink; this demonstrates the complexity of the project, and the challenges we faced.


Rover deployment switch
Rover
Custom 3D printed chassis
Custom drivetrain solution
1503 lines of code in **Python and Arduino C**
3 custom PCBs, and Computer-on-Module
Landing module
Custom 3D printed shell
Aluminium and acrylic structure
Electro-mechanical release system
Custom parachute
422 lines of code in **Arduino C**
1 custom PCB + GPS
Long-range Wi-Fi data link
433MHz radio data link
Parabolic Wi-Fi antenna
YAGI 433MHz radio antenna
Network router and access point
Antenna direction assist
Arduino Mega + TFT mini-display for operator
Computer
Mission Controller
Accelerometer and compass
Custom mobile power source
Base station
Two antennas with custom mounts (parabolic Wi-Fi and 433Mhz YAGI)
'Antenna direction assist' system with custom electronics and screen (depicted)
"Mission control" graphical user interface
3462 lines of code in **Arduino C, Java and C#**

2.2 Mechanical Rover Design

2.2.1 Design Justification

The secondary mission had scope for many different mechanical CanSat designs, which had to be individually evaluated and decided upon. This section provides a brief overview of these evaluations and the conclusions reached to give us the final mechanical rover design.

The rover itself had to be able to traverse uneven and rough terrain without getting stuck, while providing enough space internally for the electronics required. We initially considered mounting the wheels on the side of the rover, which would have given the rover two points of contact with the ground. The problem envisioned with this design was that the main body of the rover would have spun rather than the wheels, as the wheels would have been prone to getting stuck on the rough terrain. A support protruding from the main rover body was suggested, but considerations with regards to its deployment from the launch vehicle resulted in rejecting the idea of the two-wheeled rover.

The other idea considered was using tracks. The issue here was that the wheels would have to be mounted along the width of the rover, meaning that the motors used would have to be quite small. Additionally, there would be less space inside the rover for electronics. After researching the possible motor choices and conducting initial CAD modelling described below, we concluded that because of the increased stability of the rover in this configuration on the uneven terrain, the track idea was the most reliable solution.

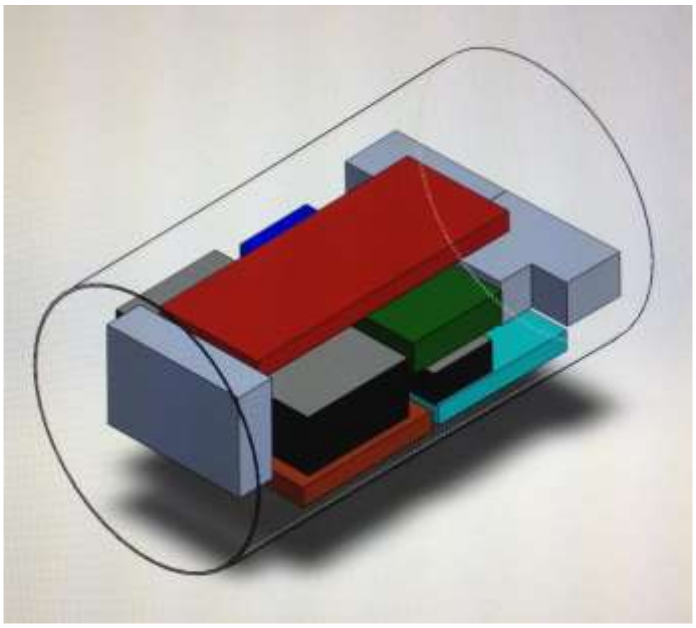
2.2.2 3D CAD Modelling

The hardware modelling and prototyping encompassed two major revisions of the 3D CAD design. Throughout the following section, these are referred to as the first and second revisions.

2.2.2.1 Component Placement

During the initial design phase, we used SolidWorks to establish how the components would fit inside the rover. This was done with careful collaboration with the systems team, who created a spreadsheet containing PCB dimensions, and describing their ideal respective positions within the rover. This process led to a deeper understanding with regards to how much spare space we would have in the vehicle.

For the initial design, before the second revision of PCBs, the component placement was very tight, as seen in the image below. Throughout the design process, the PCBs were modelled as blocks to ensure that we allocated enough space for all components.



Following the PCB revision, significantly more space was made available, due to the altered dimensions of the PCBs. Overall, all PCBs were made smaller, other than the CPU board, which was small enough and complex to assemble, due to fine pitch components, so did not justify a redesign.

In the most recent version, an image of which can be found in *section 2.2.2.2*, the camera module faces the front of the vehicle to avoid the use of any mirrors, which would have complicated the design. The batteries must be placed close to the power PCB, as well as near an access port to allow for easy removal for charging. The CPU and Gumstix are integrated as one unit - the Gumstix slots onto the CPU. As this device heats up significantly during mission operation, it should not be below other circuitry. The MCU PCB should be situated close to the CPU. Further, following the PCB revision, it was made smaller in all dimensions; however, its width did not allow it to be placed parallel with any other components. As such, it was positioned with its longest side perpendicular to the rover driving direction, meaning it required a significant proportion of the width of the rover. After these were fitted in, there was enough room for the GPS module and WiFi module next to the Gumstix. The motors were then integrated with enough space around them, allowing for the driver wheel to go over the main body of the motor. The idler rollers fit in three of the four corners, with the other corner housing the driving wheels and motor. Three pins go through the idler roller corners the entire width of the rover to keep the idler rollers in place and to secure the two halves of the can together. This is described in more detail in *section 2.2.1.3*. The chassis is split in two halves, allowing for easier assembly.

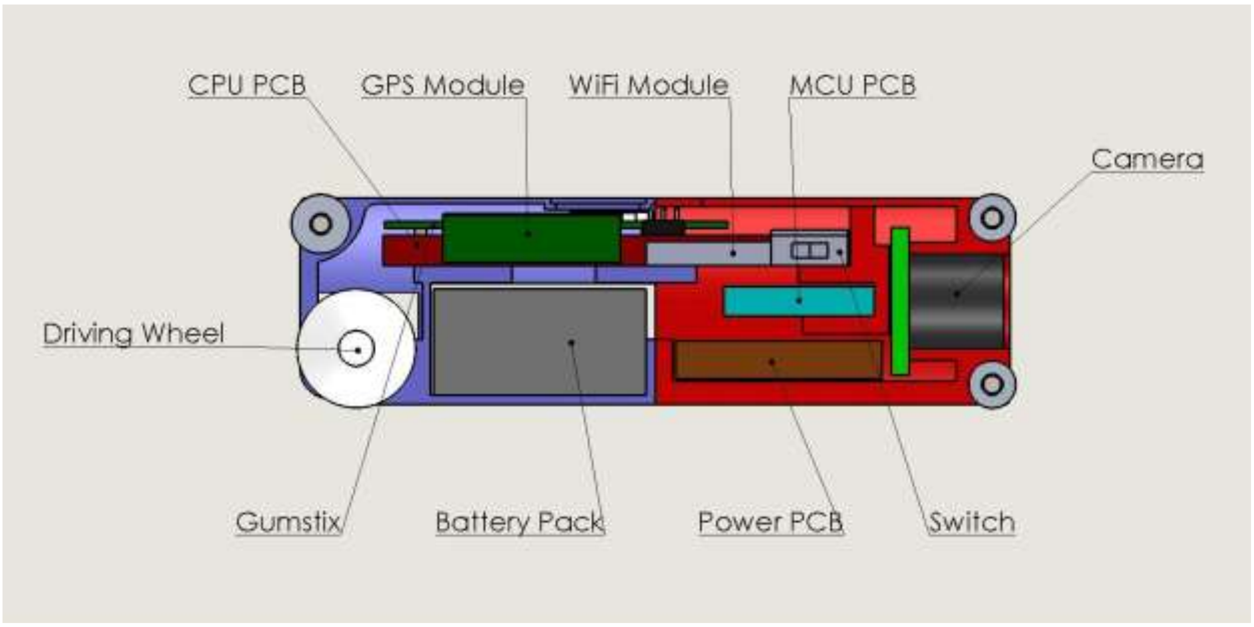
This positioning is based on the second revision of PCBs, and did not changed significantly in each revision. The only components that were given consideration to in later revisions included the USB WiFi module, which fit next to the GPS module with plenty of space to spare, and a power switch.

2.2.2.2 Chassis and Internal Support Structures

The components within the rover are held in place by a series of internal support structures, which also serve as the internal skeleton of the car, preventing significant damage, should the landing involve a higher than expected deceleration upon impact. During prototyping, we made minor developments between revisions. The structures supporting the MCU and GPS PCBs no longer extended to the entire width of the vehicle. The CPU support was split into two platforms instead of one. Both these modifications allowed for easier wiring between PCBs upon final assembly. The camera PCB supports were made to allow for more room, as it was noted that the camera did not fit in as nicely as expected.

These structures are incorporated into the external chassis - called the shell - of the rover. This option was chosen over separate discrete support parts within the shell for ease of assembly. The entire shell structure could be 3D printed as two units, allowing for easy integration of components.

Below is a labelled cross-section of the rover.



2.2.2.3 *Other Modifications*

Following further analysis of our first prototype, we noted that the two bolts to hold the two halves together would not be the most refined solution, as there was significant difficulty in separating the two halves quickly for planned and potential emergency servicing. A new solution was needed.

The holes for the bolts were removed, which allowed for significantly more space within the rover. Further, the internal covers for the idler rollers were also removed, due to their perceived redundancy in the design. The interlocking teeth that were originally incorporated were deemed to be too weak for our purposes, given the forces expected upon impact. There was no space vertically to accommodate thicker structures, so these were removed, which allowed for the support structure to be separated, which will assist in wiring the PCBs to each other.

Our solution involved using the three pins on each corner of the structure. It was discovered during assembly of our first prototype that these alone were enough to keep the parts together. The issue was that we had no method other than glue to keep these in place. The solution was to use M3 threaded rod, instead of 2.5mm copper rod, as these structures. On each corner, there would be an

M3 nut securing these in place. The required modifications were made to the CAD model, by indenting an area around the axle hole for the nut to prevent the can exceeding the 66mm diameter limit.

Should servicing be required, only three nuts need to be undone to allow the two halves to completely split.

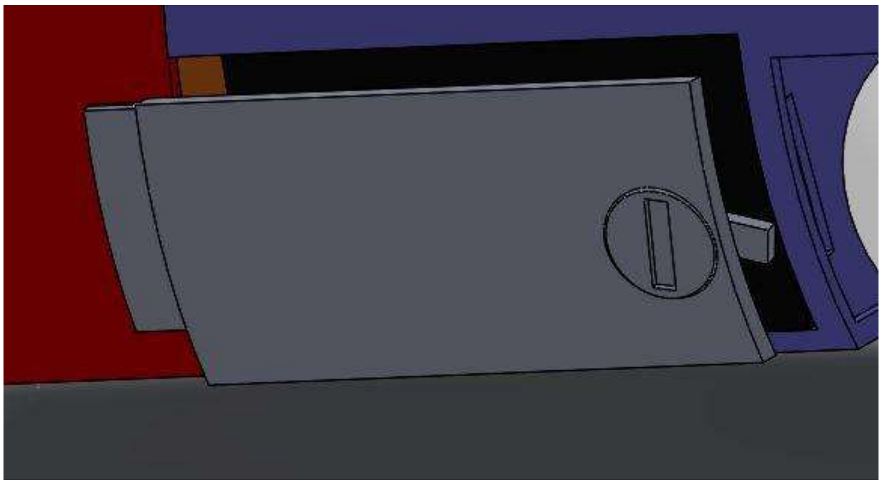
Further minor modifications were also made to the motor holders, idler roller diameters and SD card access slots following observations made in the first prototype.

2.2.2.4 Final Assembly Method

Each half, the idler rollers, the battery cover and the driver wheels are 3D printed on the HP DesignJet 3D Printer. The support material needs to be removed by a 10-hour wash in the HP removal system.

As described above, once the PCBs are wired together, tested by the systems team and the motors installed, the two halves are assembled with three pieces of M3 threaded rod and six M3 nuts.

The battery cover is assembled separately from the main unit, using glue to attach the latch onto the rotating pin, as these two components must be able to move independently from the cover in order to lock the cover in place on the rover. Once this is completed, the battery can be inserted and the cover installed.



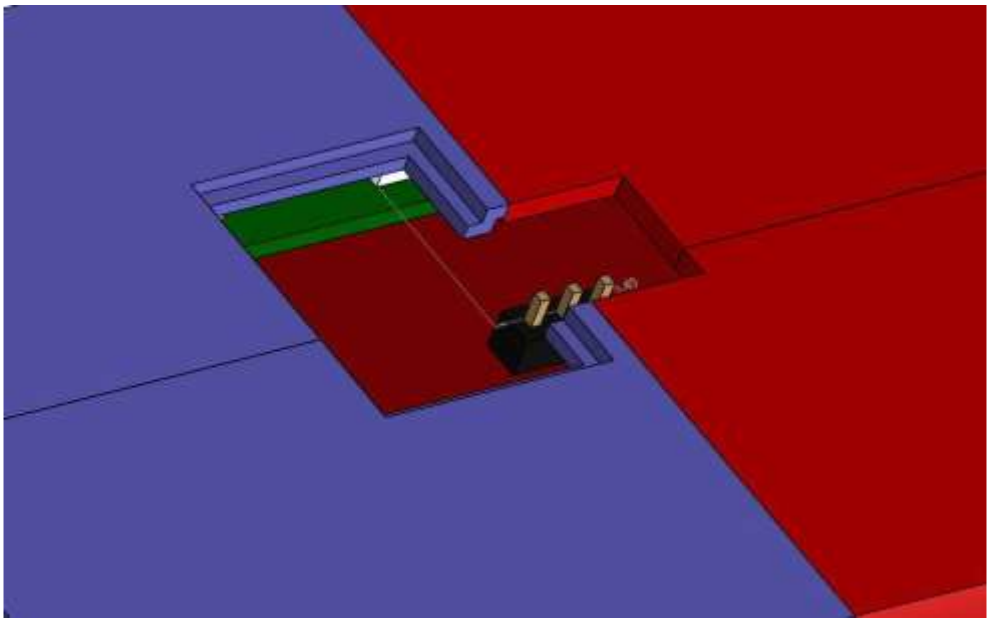
2.2.2.5 Access Considerations

It was brought to the attention of the hardware team that there were a number of considerations that needed to be implemented into the second major revision of the model to ensure functionality and quick maintenance abilities.

The team would need access to the batteries to recharge them, as the heat generated by them during this process would shorten their useful lifetime if not properly ventilated, while also potentially damaging other components. A hatch was incorporated into the design as this would allow for immediate access to the batteries, while also keeping them in place when the rover is operational. The hatch is opened by rotating the small, free-moving latch, and pulling the cover outwards. The batteries then easily slide out.

This was further modified in the second revision, as the hatch was too large when printed.

The systems team also brought up the need to have immediate access to the programming interfaces and SD card slot on the CPU board and Gumstix respectively. As these two components are connected together and are located on the top of the rover to facilitate heat dispersion, it was only necessary to create extrusions on the top of the rover. These are positioned in such a way as to be covered by the tracks while the rover is in operation, which prevents debris entering the vehicle. Access to these components aids setup, testing, debugging and data recovery.



Both holes also have a 1mm rim around their edges, which would allow for a 1 mm acrylic plastic covering, should it have been deemed necessary. A mouldable polymer - such as “Polymorph” could have also been used to serve this purpose.

2.2.2.6 Manufacture

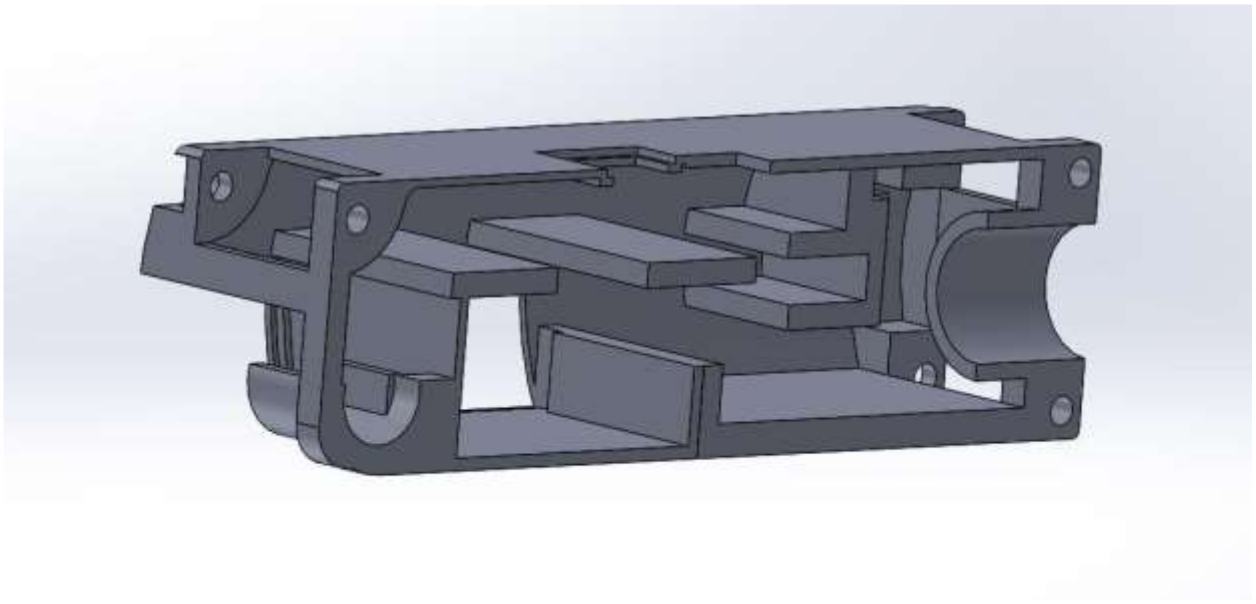
The model was designed to be printed on a Fused Deposition Modelling (FDM) 3D printer. We have two printers at our disposal: the RepRapPro Huxley and the school-owned HP DesignJet. The former provided a very easy way to prototype simple structures, such as the basic shell structure presented in progress report 1; however, due to the lack of dedicated support material, printing accurate models on it became more challenging. As the updated shell and rover designs included more intricate structures, the team decided to invest in printing the model on the HP DesignJet printer, which provided soluble support material printing capability. The Huxley was still be used for prototyping smaller items.

For a cost breakdown of all mechanical parts, including 3D printed components, please see *Appendix C.2*.

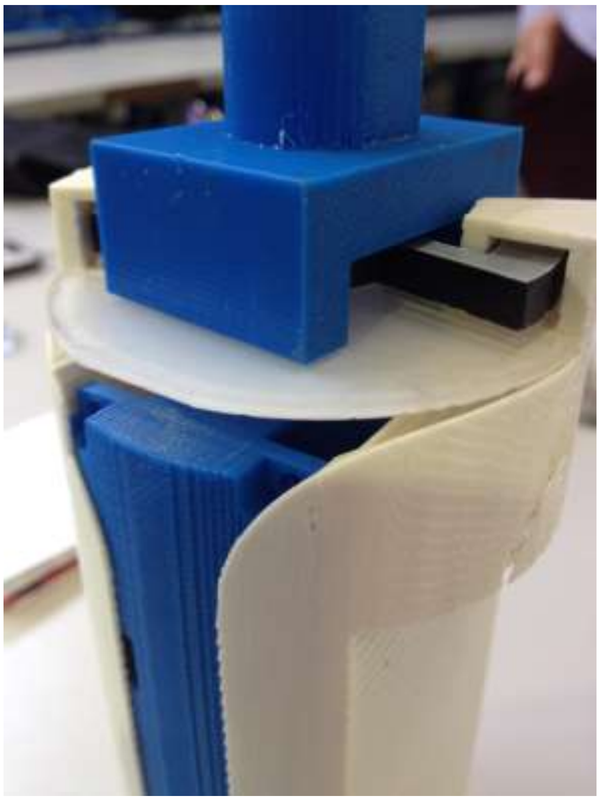
Due the manner in which the files were converted to .STL format - the accepted format for the HP DesignJet software - the team encountered issues with printing the first model. The print head would often jam, while the model would warp during printing due to a blockage in the flow of extruded filament.

We attempted the print of the first model several times, which delayed our progress somewhat. The issue was rectified and assembly of the first prototype commenced the week of the 25th November. Following the evaluation and redesign phase, the printer failed again. It took two weeks to rectify the situation.

Below is an image of the left half of the rover.



Dry fitting the rover together with the landing module shells was an important step in determining whether dimensional inaccuracies were still present in the second revision. This was conducted as soon as possible after all the components were 3D printed.



2.2.3 Rover Tracks Development

2.2.3.1 Rubber Bands

Development on the track design for the vehicle began shortly after evaluating the first prototype with regards to PCB placement. Our initial idea was to use rubber bands, as this would provide the necessary tension around the vehicle and friction to the driving wheel, also covered in rubber band.

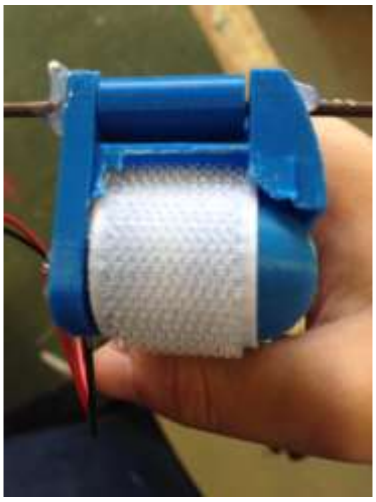
We discovered several problems with this method:

- We could not source rubber bands of the required length and width, which meant that we had to glue fragments together
- The wheel would slip or jam completely. Smooth driving could not be achieved

- The rubber caught on the base and top of the vehicle

2.2.3.2 Velcro

The second idea was the use of Velcro strips as the track (hook side) and on the wheel (soft side). Modifications had to be made to the chassis to allow space for the driving wheel covering. These changes were implemented in the second CAD model following testing.



We discovered issues with this method as well:

- The velcro stuck too much. No smooth driving could be achieved
- There was no tension in the track. It would slip off the vehicle
- The idler rollers did not assist. They were points where the track slipped

2.2.3.3 Final Solution

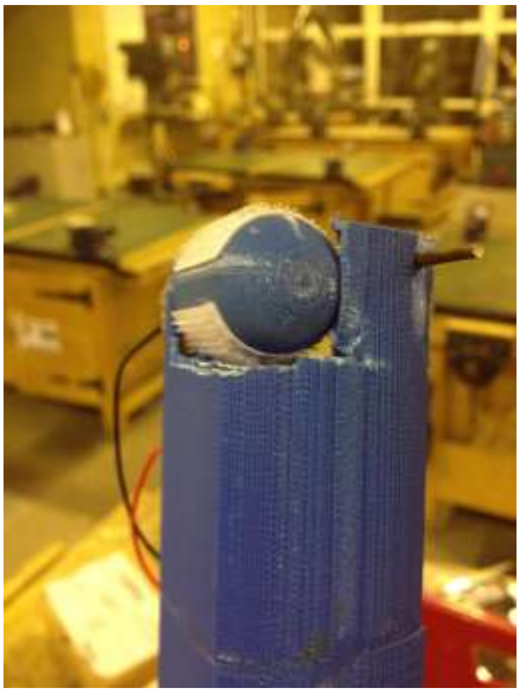
Taking the lessons from the first two tests, we developed a set of criteria for the track system:

- The track must be tensioned
- The track must not slip off the vehicle
- The method by which the track is driven must be able to be driven by the strength of the wheel coupling
- The track and wheel material must not stick, but rather provide adequate friction
- The track material must not have too much friction against the chassis.

Our solution was a hybrid of the two ideas presented above: the track material would be made of Velcro (hook-side) strips, joined together by short bits of rubber band. This provides the required length and tension, while slipping smoothly along the grain of the FDM 3D printed chassis.

The driver wheel is wrapped in Velcro (hook-side) material as well. This, we discovered, when coupled with the track described above, results in high torsional friction between the two surfaces, but low resistance when pulling them apart.

All six idler rollers will also be coated in Velcro (hook-side) material to prevent slipping at these locations.



2.3 Rover Electronics design

2.3.1 Electronics Design Justification

The primary mission naturally involved measuring temperature and pressure at frequent intervals. This necessitated the inclusion of a temperature and pressure sensor. In order to save space, which was at a premium, we chose a combined temperature and pressure sensor – the MPL3115A2.

We decided to add a camera because streaming live video will make driving the rover from a distance easier, and to make the project closer to a real space mission, which the CanSat aimed to simulate. We also added GPS to make locating the rover easier; this also meant that it could travel to a location automatically.

Many of the objectives did not necessitate single measurements: they were aspects of our project, which needed to be tested for functionality, but not necessarily measured directly. This was because our secondary mission was concerned predominantly with deploying a vehicle rather than measuring the surrounding environment. However, the list of key sensors on board the CanSat and its recovery systems are as follows:

- Combined pressure and temperature sensor (x2)
- CMOS camera
- GPS (x2)

Where sensors are marked as x2, that means one was present on board the rover, and one was in the landing module. The landing module is described further in section 2.5, and its electronics in section 2.5.6.

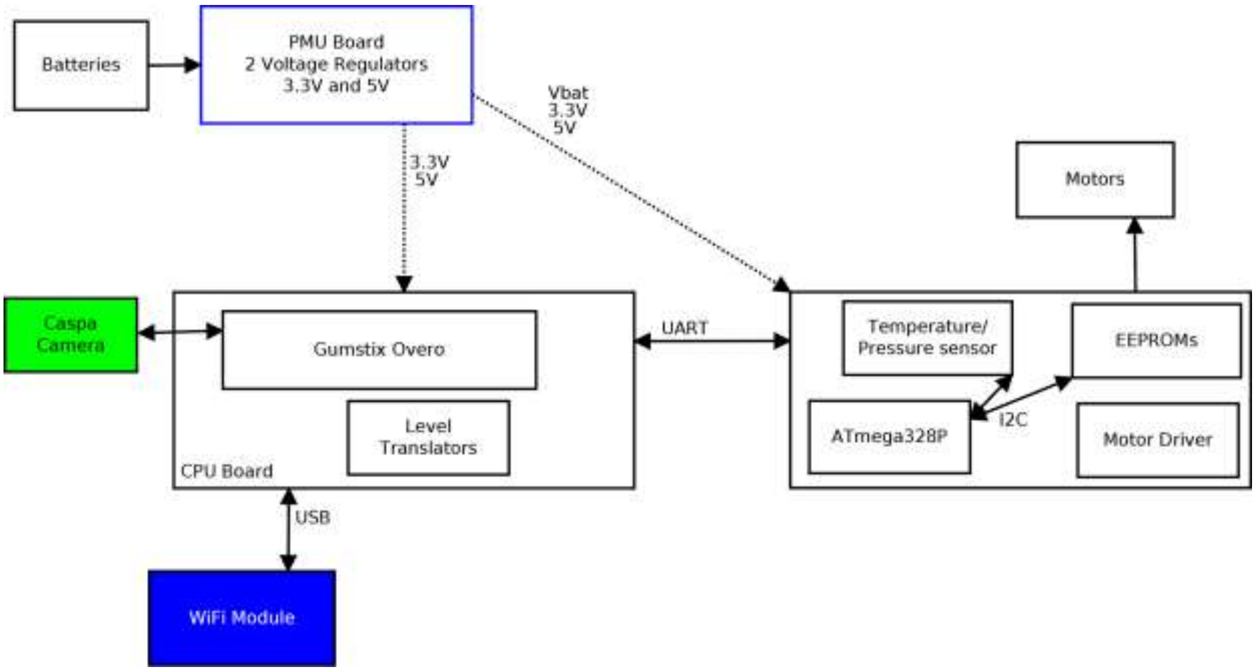
2.3.2 Electronics Overview and Block Diagram

There were a number of constraints on the electronics that meant the design was not as straightforward as many other projects. The particular limit is size, as not only did the electronics have to fit into the rover, but the batteries, motors and mechanical supports did too.

This meant that choosing a processor board that could handle overall rover control, navigation, and compression and streaming live video was a challenge. The most obvious choice was a Raspberry Pi, however, this would not fit when the size all of the other components was taken into account. Considerable research was put into finding a solution that would be ideal, and eventually two possible options were deemed feasible - the Variscite DART-4460[2] and the Gumstix Overo TidalSTORM[3]. We decided on the Overo TidalSTORM because the DART-4460 could only be purchased as part of a very expensive development kit, whereas Gumstix agreed to sponsor us by providing the hardware for free.

All electronic designs can be found in *Appendix* B. In addition, *Appendix C.1*, contains a full cost breakdown of all electronic components and modules.

Below is a simplified block diagram of the electronics.



Before a detailed description of each section, here is a brief overview of all key components.

2.3.2.1 Power Supply

This section is the most important, and often most forgotten, part of any piece of electronics design. Its precise details of operation will be described in the 'Power' section that follows, but for now a brief function specification shall suffice. The power supply has an input from two 18500 size lithium-ion batteries (nominal 3.7V each), and outputs 2 voltage rails:

[2] See: http://www.variscite.com/products/system-on-module-som/cortex-a9/dart-4460-cpu-ti-omap-4-omap4460 [accessed 20/08/2013]
[3] See: https://store.gumstix.com/index.php/products/356/ [accessed 20/08/2013]

- 3.3V, for the Gumstix, surrounding electronics as well as the temperature/pressure sensor and EEPROMs on the MCU board.
- A 5V rail for the USB WiFi module and Arduino microcontroller.

The original version of the power supply board also had a secondary 5V rail, originally intended for powering RC servos, which would be used to deploy the rover; however, it was later decided the rover deployment mechanism would be part of the landing module (again, this will be mentioned later). This was removed in revision B to save space.

Power supplies to the various boards are indicated by dotted lines on the above diagram.

2.3.2.2 CPU Carrier

The CPU carrier is responsible for breaking out the signals from the Gumstix and handling logic level translation (because the Gumstix requires 1.8V signals and other parts of the system require 3.3V and 5V signals). Its main input and output connections can be summarised as follows:

- A USB 2.0 link to the WiFi module (which is a repurposed USB WiFi ‘dongle’)
- UART 1, and some control signals, which connect through a level translator to the MCU board. These handle the command interface to the MCU.
- UART 2, which, through a level translator, receives position information from the GPS module in NMEA 0183 format.
- UART 3 provides a debugging interface, used to access a serial console enabling configuration, diagnostics and testing. For ease of setup and maintenance, it was decided that this port will be accessible from the outside of the rover.

The camera is the official Gumstix Caspa camera. This is connected via the Gumstix’s camera interface - the connector for this is mounted on top of the Gumstix itself, so the signals are not routed through the CPU board. We chose this camera, as a range of software for the Gumstix and community support was available. Due to bandwidth and processing power limitations, ‘video’ streaming will be implemented as a static image that updates 2 or 3 times a second.

We chose WiFi as our communication protocol of choice as more bandwidth is required than is available from other simpler systems in order to transmit images. Additionally, WiFi equipment was easily available at a relatively low cost.

2.3.2.3 MCU board

The heart of the MCU board is an ATmega328P (in the TQFP package) running the Arduino bootloader. The Arduino board is primarily responsible for the parts of our project which require a simple, real time processor. It receives commands from the Gumstix over a serial interface. It is responsible for the following tasks:

- Failsafe temperature and pressure logging: the temperature and pressure sensor (a MPL3115A2) and two 24LC1025 EEPROMs are connected over the I^2C bus to the ATmega.
- Motor control: this board has a two channel motor driver for the main motors.
- The data link to the landing module, described in *section 2.5.6.2*.

2.3.3 Power

2.3.3.1 Batteries

The system will be powered from two 18500 size lithium-ion batteries. This size was chosen as it was small enough to fit in our proposed design, but available with relatively large capacities – research showed that 18650 size batteries were available in considerably larger capacities, but would have been too large for our proposed design. We used 1600mAh batteries, at a nominal voltage of 3.7V each and a useful voltage range of 3.4-4.2V. This gave an approximate energy of 5.9Wh, or 11.8Wh in total.

During electronics testing, measurements of the current consumption were made (the currents varied quite significantly which limits the accuracy of the readings to 2 significant figures). These readings were made at the nominal voltage of the battery pack, 7.4V. This figure was chosen, as at this point approximately half the energy will have been transferred, therefore the current values can be used to calculate battery life accurately. Due to the use of switching regulators in the power supply section, the power (but not current) consumption from everything other than the motors will have remained relatively constant as the battery discharged.

The figures have been updated to reflect the most recent hardware revision, using the latest available software version.

State	**Current /mA**	**Power /W**
Wireless off, camera off, motors off	160	1.2
Wireless on, camera off, motors off	230	1.7
Wireless on, camera streaming, motors off	290	2.1
Wireless on, camera streaming, motors on	560	4.1

CanSat requirements state that it must be able to be left on for 3 hours before launch. During this time, we expect WiFi to be connected, but no video to be streaming. That means that there will be (11.8 - 1.7 * 3) = 6.7Wh remaining, enough for (6.7/4.1) = 1.6 hours of motor control. We expect this to be more than sufficient time.

The exact batteries we used were Torchy brand, as tests by other users show that they met their true 1600mAh capacity, whilst not being overly expensive. We also purchased a charger, as the batteries were charged outside of the rover to ensure sufficient ventilation and cooling. For safety reasons, all battery charging took place on a heatproof mat in the unlikely event that the batteries overheat or ignite.

2.3.3.2 Power Electronics

Please refer to the schematic diagram and PCB layout in Appendix B.1.

The first responsibility of the PMU board is reverse battery protection. This is implemented with an N-channel MOSFET between the battery negative input and 0V rail, with the gate connected to V+. In this configuration, no current will flow unless the battery is correctly connected. This is designed to prevent against a simple careless error causing a catastrophic failure.

In addition to this, there are two voltage regulators, each of which is a ST L5973D[4]. This is a 2.5A switching regulator, and is used in 'buck' mode, where it steps down the voltage (because each of the required rails is below 6.8V, the minimum voltage of the batteries). We could have used a conventional linear regulator (or LDO), but a switchmode design was chosen as it is more efficient (typically 90% or greater). An additional advantage is that, because of the higher efficiency, very little heat is produced, simplifying thermal design.

These regulators have their output voltage set using a potential divider supplied from the output voltage rail, with their output into the 'FB' pins of the regulators. These regulators are designed in such a way that they try and keep the FB pin at 1.235V, so the resistor values were chosen to output 1.235V when the regulator output is at the correct value. These are resistor pairs R2/R3, and R6/R7.

Due to the relatively high frequency and current levels at which the regulator was operating, good PCB design was important. We ensured that there was a solid ground plane throughout, ideally on both sides of the board (this also helps dissipate the small amount of heat produced). Traces were kept as short as possible, and all high current traces, as wide as possible to lower their resistance. To keep ground plane resistance low, and prevent 'ground bounce' during switching, vias connect the top and bottom ground planes where they are interrupted by other traces.

High quality tantalum capacitors were chosen for the output decoupling capacitors C3, and C7, as these parts are critical to the reliable operation of all of the other subsystems. Low ESR capacitors were chosen to ensure reliable, noise-free output and keep the regulator's control loop stable. Higher capacity ceramics where evaluated, but had too high an ESR and have issues with capacitance decreasing as DC voltage increases. Surface mount electrolytics were too high and would have meant the board would not fit within the available space constraints.

The 2 LEDs provide an indication that the voltage rail is powered. This was useful for debugging to ensure that the regulator is functional and not overloaded or defective.

On the Rev A board, there was a '5VS' voltage rail, originally designed to provide a separate supply for any servos onboard the rover, as servos would produce a large amount of electrical noise on the supply rail. This is not present in the Rev B board to save space and reduce quiescent power draw.

[4] See: http://www.st.com/web/en/resource/technical/document/datasheet/CD00002851.pdf [accessed 21/08/2013]

2.3.4 Gumstix and Related I/O

2.3.4.1 The Gumstix

The Gumstix Overo, pictured above, is a small 'computer-on-module' that runs Linux on an OMAP3 ARM processor. We used the TidalSTORM variety, which has 1GB of RAM, a DSP and hardware video acceleration. The combination of a large amount of RAM and hardware video means that we can stream video efficiently, reliably and with surplus CPU power for the other control functionality.

2.3.4.2 CPU Board

Please refer to the schematic diagram and PCB layout in Appendix B.2.

The primary purpose of the CPU board was to provide connections at the correct logic levels for the other devices in our project. J1 and J4 are the two connectors that the Gumstix plugs into.

The Gumstix accepts an input voltage of approximately 3.3-4.2V. We decided to use 3.3V, as this voltage is also required elsewhere so this would cut down on the number of voltage regulators required. The Gumstix performs without any problems at this voltage.

The two other significant devices are U1 and U2. These are TI TXS0108E bidirectional logic level translators[5]. These convert from the 1.8V I/O of the Gumstix to the 3.3V/5V I/O used elsewhere. These level translators were chosen as they support the voltage levels we are using. The bi-directionality also makes them very easy to implement, as they will automatically sense whether each pin is an input or output, removing the need for any configuration. Another important feature these devices have is an output enable pin (OE). This pin is connected to the SYSEN output from the Gumstix. Due to the nature of the processor on board the Gumstix, it is pertinent that none of the Gumstix pins are driven high or low until this pin comes high, otherwise damage could occur.[6]

U1 converts from the 1.8V domain of the Gumstix to the 5V domain for the connection to the Arduino microcontroller on the MCU board. U2 converts from the 1.8V domain to the 3.3V domain for the GPS module, and the debugging interface.

Attached to the Gumstix is a GPS module. We chose GY-NEO6MV2 GPS module as it is relatively small, and has a detachable active patch antenna to ensure accurate tracking. It also has a built-in supercapacitor back-up so will obtain a fix almost instantly provided it has been previously powered

[5] See: http://www.ti.com/lit/ds/symlink/txs0108e.pdf [accessed 22/08/2013]
[6] See: http://www.gumstix.org/hardware-design/overo-coms/73-overo-design/199-overo-expansion-board-design-requirements.html [accessed 22/08/2013]

on in the last few days. The GPS module also enables the Gumstix to determine the date and time without the need for a hardware real time clock, as GPS can provide a very accurate UTC reference.

The debugging interface is UART3 of the Gumstix. This port defaults to acting as a Linux console, and allowed the system to be tested, configured and developed on when a wireless link is unavailable, as well as providing troubleshooting in event of a system failure. Any standard USB-UART adaptor can be used. For development the Dangerous Prototypes Bus Pirate was chosen as it operates with 3.3V I/O levels. The systems team liaised with the mechanical team to ensure this connector could be accessed from outside of the chassis.

IC1 is a 1.8V voltage regulator (MCP1703-1802E). This provides a 1.8V reference that the level converters use for connection to the Gumstix.

The CPU board was tested, and all key functionality worked. As the design could not be made significantly smaller, and there were no corrections or feature changes necessary, another revision of the CPU board was not made. This also saved considerable time, as the CPU board was relatively difficult to assemble due to the large number of fine pitch components.

2.3.5 Arduino Microcontroller

Please refer to the schematic diagram and PCB layout in Appendix B.3.

The MCU board contains an ATmega328 processor[7], two 24LC1025 EEPROMs[8] and a TI DRV8833 motor driver[9]. This microcontroller was chosen as there is a large amount of community support available for it, due to the popularity of the Arduino environment. It is also an ideal size and has a sufficient number of pins for our application.

The DRV8833 motor driver was chosen for a number of reasons. It was available in a TSSOP16 package, which is small enough to fit inside our space constraints, but still easy enough to hand solder. It also has configurable current limiting, which will have prevented damage to the motors if they stalled or short-circuited, as well as over-temperature protection. The current limit is set with R20 and R21. All of the motor control pins have been connected to PWM outputs of the Arduino, to provide the ability to control the speed of the rover. It also can handle the voltage supplied by the batteries (8.4V), so no step-down converter is needed for the motors.

This board connects to the CPU board with 4 wires: MOSI, MISO, RST and RDY. MOSI and MISO form a serial link, with MOSI ('Master Out Slave In') going from the CPU to the Arduino, and MISO ('Master In Slave Out') going from the Arduino to the slave. The RST line allows the CPU to reset the Arduino if it locks up, and is also required for reprogramming the ATmega using the Arduino bootloader. In Rev B of the boards, it is connected through a capacitor to the CPU board so the ATMega will only reset on a pulse from the Gumstix, preventing the Gumstix holding this line low and keeping the ATMega in reset - defeating the primary mission failsafe described below.

[7] See: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf [accessed 24/08/2013]
[8] See: http://ww1.microchip.com/downloads/en/DeviceDoc/21941E.pdf [accessed 24/08/2013]
[9] See: http://www.ti.com/lit/ds/symlink/drv8833.pdf [accessed 24/08/2013]

The RDY pin functions as a form of flow control. It is an output from the Arduino, and goes high when the Arduino is ready to receive commands, and is low if it is busy processing a command (to prevent the serial buffer overflowing), or the Arduino has not finished starting up.

The final role of the Arduino is to handle the data link to the landing module. This is one of the new connections introduced in revision B of this board.

2.3.5.1 Primary Mission Failsafe

The EEPROMs and temperature/pressure measurement module both connect to the Arduino through the I2C bus. This allows the Arduino to perform standalone temperature and pressure logging if the Gumstix is not correctly functioning. This may occur in event of a software crash, but also if power is temporarily lost during descent, as the Gumstix may take up to 45 seconds to start but the Arduino will start within 3 seconds. Revision A used an external module based on the BMP085, but this added too much height. This module also appeared to have an accuracy issue, and was not purchased from an authorised source, so specifications could not be guaranteed. For the revision B board, we used a MPL3115A2 sensor, directly mounted onto the board. This was attached to the board with hot air, as it is a leadless package. This sensor has a specified worst case absolute accuracy of +/- 0.4kPa and +/- 1 degree Celsius. It is also designed to resolve relative altitudes down to 30cm in 'altimeter mode'.

We decided to do this as completion of the primary mission was very important, and it was simply impossible to guarantee high reliability of the Gumstix and the software running on it, as it was running a full operating system with many pieces of software running in the background. The Arduino is a much simpler system, which ran simpler software that could be thoroughly verified. The Arduino is also more robust to external faults, including out of tolerance power supply voltages, electrical noise on the power rails, and input voltages above the supply voltage compared to the Gumstix, so will hopefully continue working even in the event of a failure upstream.

Both software and hardware for this failsafe fully worked during testing.

2.3.5.2 Motor Choice

An important part of our project is the choice of motor, as a large amount of torque is needed to ensure the rover can handle all types of terrain; however there are strict constraints on size and power consumption.

We evaluated three types of motor under a number of criteria, including speed, torque, efficiency, size and ease of integration. These three types where chosen to be evaluated after initial research, and we would perform our own tests on each to decide which one would actually end up driving the rover's tracks.

2.3.5.2.1 Pager Motor

This is a very small motor similar to that used in mobile phones for the vibration alert. Although small, it lacks any internal gearing and ran too fast with insufficient torque. If we had chosen this motor, we would have had to add a gearbox. We could not find one available that was small enough, so evaluated 3D printing a custom one, but decided it would be too weak. Therefore, we decided not to use this motor.

2.3.5.2.2 LBD Miniature DC Geared Motor

This motor is in a form factor similar to that of a miniature servo, and has an integrated gearbox. Although tests showed it had sufficient torque and speed despite consuming a relatively small amount of power; it did not fit well in our proposed design. In addition, because it used plastic gears, testing showed it would not survive multiple stalls or heavy loading which could have occured during testing. We did not choose this motor.

2.3.5.2.3 40 RPM Short Shaft Motor

This was the final motor we tested. It offers a greater amount of torque compared to the above motor, despite consuming very little power, although it rotates more slowly. We found it to fit better into our design, and the metal gears make it significantly more robust than the previous motor.

This is the motor we used in our final design to power the rover’s tracks.

2.4 Software design

2.4.1 Rover Software Overview

There are a number of parts to the rover software. These include the code that runs on the Gumstix, which must handle wireless communication management, video stream control, and navigation; and the code that will run on the Arduino, which must handle motor control and temperature/pressure logging to the EEPROM on the Arduino board.

There are two different network systems in use. Images are served over HTTP, with a script that updates the file and basic web server providing them from the Gumstix. Control and data telemetry occurs over a custom protocol based on raw TCP/IP sockets. Both HTTP and this protocol run on top of the WiFi link.

Communications are always initiated from the ground station PC, using a carefully specified protocol drawn up to prevent any interfacing issues between the two systems. The Gumstix then handles the command as appropriate. If the command is a motor control command, the message is simply passed from the Gumstix to the Arduino. If the command is a request for temperature/pressure data, the Gumstix requests this data from the MCU board and forwards it back to the PC. GPS requests are handled directly by the Gumstix, as this is what the GPS module is connected to.

As well as a set of commands, we also drew up a set of error codes that are returned in case of a problem to aid in diagnosing the cause of failure.

The commands which we used are:

Code	**Parameters**	**Description**
`CL`		Returns the current GPS location, as well as a boolean specifying whether or not GPS is available.
`MV`	Forward, backwards, left or right	Manually move in a specific direction
`ST`		Stop moving
`TP`		Returns current temperature/pressure data
`GT`	Latitude/longitude	Start navigating automatically towards a specific waypoint
`CS`		Return any errors that have been occurred, in text form

As part of our outreach aims, all of our software is open source, licensed under the GNU GPL, and publicly available on our Github repository (https://github.com/daveshah1/nova) for the community to use and view. This also made keeping track of revisions easy, allowing us to revert to a previous, known-working state at any time.

2.4.2 Gumstix Software

First and foremost, the code on board the rover attempts to set up the network configuration to try and connect to our wireless network, the ESSID of which will be pre-programmed. Once the connection has been established, it receives control commands from the base station and returns data as necessary. These commands include navigate to a waypoint, return temperature/pressure or return current position.

Upon command from the base station, the rover's software also sets up video stream using a predefined configuration, as well as recording video locally. Originally, we expected to use gstreamer for this, as a hardware accelerated version is available for the Gumstix; however, there were many technical issues with this meaning we had to switch from a true video stream to a series of images fetched via HTTP, updating once or twice every second depending on available processing power and network bandwidth. Because the rover travelled slowly, this did not majorly affect the ability to control the rover. It also decreased the required network bandwidth significantly, enhancing performance if the wireless link is unreliable.

The final major part of the software is navigation. The rover must, when given a waypoint, take the best possible route to that waypoint, guided by GPS. In addition, if the network connection is lost, it must be able to navigate to a pre-programmed 'home' location until position is regained.

We were unable to use a digital compass, as it would have to be mounted relatively close to electronics and motors, and past experience shows these devices are unreliable under such conditions; we are thus limited to a relatively simple navigation algorithm using the GPS only. Unfortunately, a limitation of GPS is that it generally has an accuracy limited to 5-10m without very specialist equipment.

The rover first calculates the direction in which it needs to move. If it has rough idea of its current position, as it has previously moved, then it attempts to turn towards this direction. It will then move for 5-10m before checking the GPS and determining the direction it has moved so far, turning an amount proportional to the difference between the target and actual directions, then repeating the process after 5-10m until it reaches the target location. Although this algorithm is not optimal, our tests showed it worked sufficiently.

The Gumstix software is written in Python, as a large number of useful libraries are available, and a stable Python interpreter is available for the Linux distribution running on board the Gumstix.

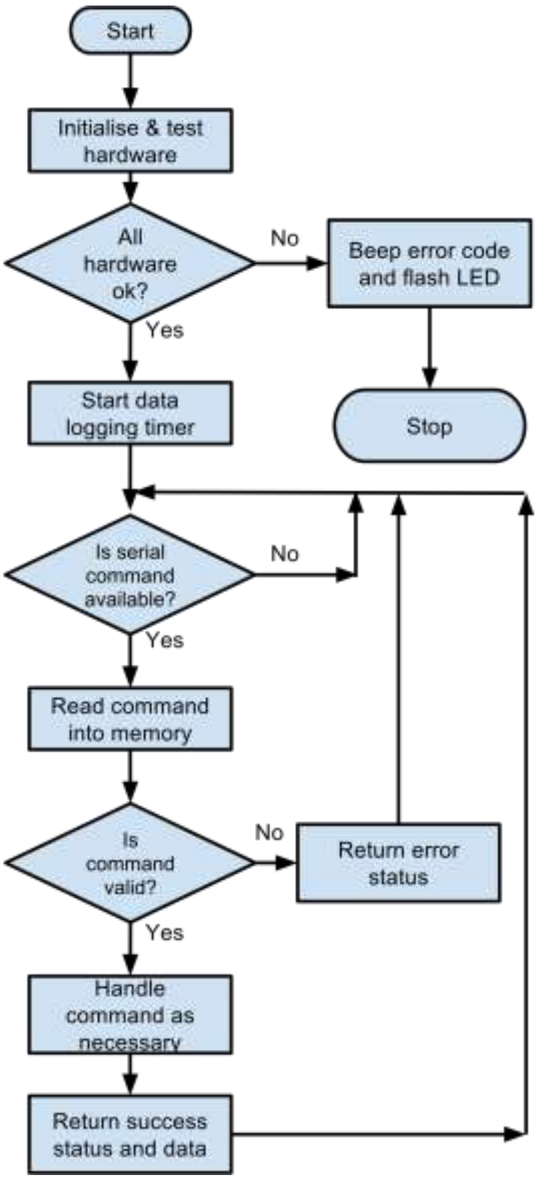
Because much of the software running on the Gumstix is open source and the fact that the community are supporting the Gumstix less, as more alternative platforms are developed, there were many issues in getting software to work, particularly with regards to the camera.

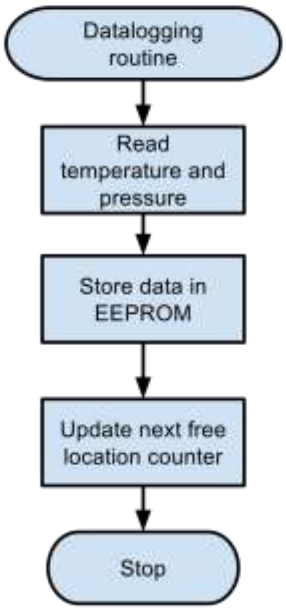
Getting images of any nature out of the camera required downloading a very old operating system image and older kernel, then cross-compiling the kernel with drivers for the camera and wireless module. This lack of support is why hardware acceleration, necessary for a smooth video stream, appears no longer to be possible, and why we had to choose using a series of images over a true, 30fps, compressed video link.

Much of the software runs as background threads, communicating using shared files. This enables data to be logged locally while the main program can wait for network communications.

2.4.3 Arduino Software

The software on board the Arduino received commands from the Gumstix over a serial link and handled them appropriately. This may involve switching motors on or off, returning temperature and pressure or reading data back from the EEPROM. In addition, some functionality was on a timer that operates at a set interval, reading temperature and pressure from the module and storing that data in the next free location in the EEPROM. A simplified flowchart of the Arduino software is below:


Start
Initialise & test hardware
All hardware ok?
No
Beep error code and flash LED
Stop
Yes
Start data logging timer
Is serial command available?
No
Yes
Read command into memory
Is command valid?
No
Return error status
Yes
Handle command as necessary
Return success status and data


Datalogging routine
Read temperature and pressure
Store data in EEPROM
Update next free location counter
Stop

The command protocol between the Gumstix is in the following form:

```
[START] [COMMAND TYPE] [OPTIONS] [STOP]
```

The start and stop bytes allow the Arduino know when a command begins, and when to stop reading the command.

The following commands were used for communication between the Gumstix and Arduino:

Code	**Parameters**	**Description**
`MT`	Two options, one for each motor - forwards or backwards	Set the direction each motor goes in
`TP`		Returns current temperature/pressure data
`RD`		Download EEPROM data
`FT`		Format EEPROM

After every command, the Arduino sends a reply with a status code. This could either be success or failure. If no reply is received, or an error code is returned, the Gumstix will re-send the command. The reply packet looks like:

```
[START] [STATUS] [DATA] [STOP]
```

Status codes will include success, command not recognised, options invalid or hardware fault occurred.

In order to allow reliable data storage and recovery, a specification has been drawn up as to how data has been stored. All storage takes place in 8 byte 'blocks', which prevent alignment issues.

For ease of processing, data that spans more than one byte is stored as big-endian - most significant byte first.

All blocks contain 7 bytes of data and a 1 byte checksum. The checksum is used to easily identify corrupted data. It is calculated as follows:

$$((7 \text{ x byte } 0) + (6 \text{ x byte } 1) + (5 \text{ x byte } 2) + \ldots + (1 \text{ x byte } 6)) \bmod 256$$

The first block of device 0 has data as below (followed by a checksum as calculated above):

```
4e 4f 56 ** ** 00 00
```

The two bytes marked as ** are used to store the location of the next free block, so that logging can continue after a power off.

Every time a new logging session is started, the following block is written to function as a separator:

```
50 41 47 45 00 00 00 2D
```

You will note that the last byte is a checksum, as described above.

Actual logging blocks are as below (again, the final byte is a checksum as described above):

```
PP PP PP TT TT tt tt
```

The three ‘PP’ blocks store pressure in Pa, as an unsigned 24-bit integer.

The two ‘TT’ blocks store temperature in °C * 10, as a signed 16-bit integer.

The two ‘tt’ blocks store a timestamp, as an unsigned 16-bit integer, which is the number of milliseconds since the last logging session.

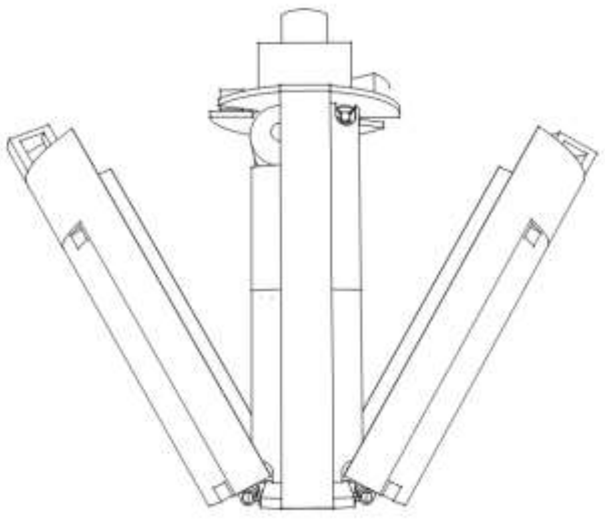
2.5 Landing Module and Recovery System

2.5.1 Design Justification

After deciding on the rover design, considerations had to be made as to how the rover would be deployed from the launch vehicle in such a way that the required parachute would not impact the ability of the rover to manoeuvre on the ground. In this respect, the design had to incorporate a way to discard the parachute after the descent. Linear servo motors were suggested and prototyped as a release mechanism within the rover itself; however, after the initial CAD modelling, we realised that the limited pace would not allow for a mechanical release system. We did notice that there was a lot of space above and below the rover – as these surfaces had to be flat. This space within the cylindrical launch vehicle could be used to house this release system. To do this a separate enclosure would have been required. Thus, we concluded that the design of a landing module would be the best solution to the issue of discarding the parachute.

2.5.2 Landing Module Mechanical Design

2.5.2.1 Landing Module Hardware



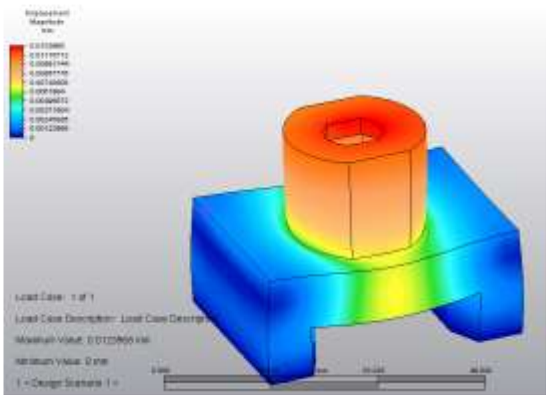
The landing module was designed in such a way as to minimise the chance of mechanical failure following impact. Furthermore, it was designed so that should there be a mechanical error, the rover would have still retain been able to some of its functionality, such as the ability to take images and record temperature and pressure data.

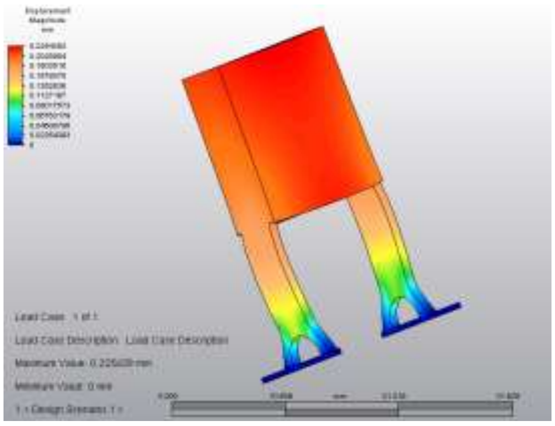
The release mechanism is designed as follows: two halves, both housing the module’s electronics and power supply, are hinged from an acrylic base plate. On top, a motor, the same type used in the rover, directly drives a latching mechanism, which keeps the two halves together during ascent and descent. There is no external gearing to reduce the possibility of mechanical failure. The rotating part of the latch is made out of laser cut acrylic, used to allow for a more precise friction fit to be manufactured into the part to make attaching it to the motor shaft itself easier.

Each half is suitably chamfered to allow for the two halves to fall outwards on their hinges when the rover is deployed without coming into contact with the top plate. One half will primarily house the landing module PCB, including the microSD card for data storage and retrieval. The other will primarily be used for battery storage. Each half of the module has a partial removable outer shell bolted in place that provides easier access to the internal electronics than a single piece would have provided, should servicing of the electronics or replacement of the batteries be required. In addition, it serves as a protective and shock absorbing layer for the internal electronics. Each half and removable shell has been 3D printed on the HP DesignJet 3D Printer alongside the rover chassis from ABS. This final manufacture was preceded by a series of prototypes produced on the RepRapPro Huxley.

Each half will also have a "hook" integrated into the top of the part that will form part of the latch based release mechanism described in more detail in *section 2.5.1.3*.

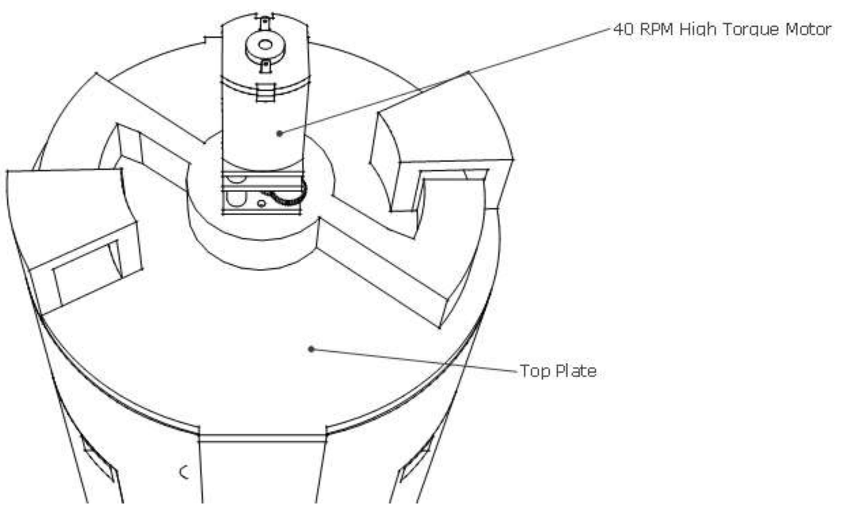
2.5.2.2 Motor Mounting Components





We suspected that the old motor holder design did not hold the necessary strength to withstand the necessary forces upon initial development. Since this holder must have maintained the position of the motor closely in order for the release mechanism to operate once the can was on the ground, it was paramount that the part did not deform too much on impact. Thus, we conducted stress tests on the old design and on the new design that we proposed – a much more substantial part with a stronger support structure. Applying an arbitrary force in a simulation to both designs from the top down illustrates the advantage of the newer design, which we manufactured used in the final can assembly. In places, the deformation under stress in this scenario is up to twenty times less in the new design that in the old one. In other words, the new design is twenty times stronger than the old.

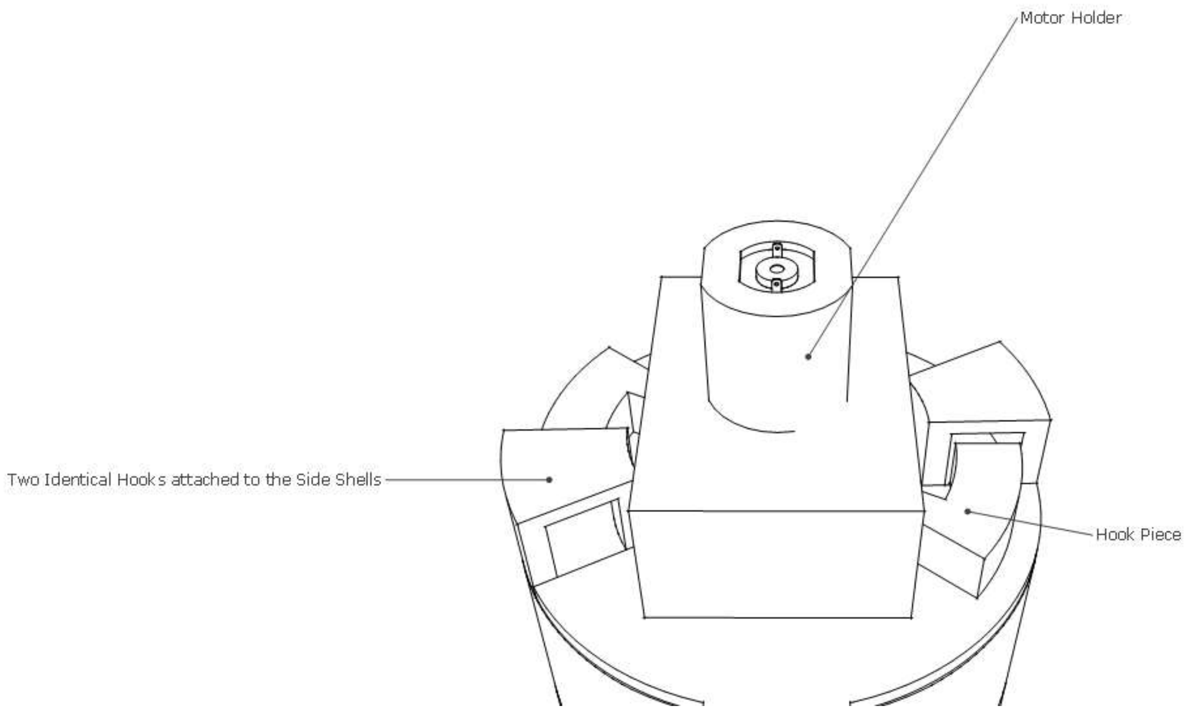
Hence, we progressed with the second design to hold the motor in place above the latch mechanism (which will be described in more depth in the next section). The 3D printed motor holder keeps the motor in position above the geometric centre of the can and of the mechanism itself. It has two large supports which provide more support for the holder while allowing for enough space for the latch to rotate through the required angle. The motor itself is very tightly friction-fitted into the holder; it is not removable by hand alone, and thus is very unlikely to move or come loose during the descent or, more importantly, on impact when the loads on the holder will be at their peak.



2.5.2.3 Latch Mechanism

As mentioned in an earlier section, each side shell has been designed specifically to integrate a "hook" into the top of the part. On the top plate of the overall landing module (and can) a latch mechanism, driven by a 40 RPM high torque motor identical to the one used in the rover to supply the drive, has been integrated. It is made from 7mm thick acrylic, used partly for its strength compared to the 3D printed ABS used elsewhere within the landing module.

This motor is activated, releasing the latching mechanism by rotating it out of the integrated hooks, causing the module to unfold. The two side shells are sufficiently weighted that they should open freely upon this event. If this become an issue during later testing then slight spring loading could have been implemented without compromising the strength and so reliability of the latch system. This was not required in the end. See s*ection 2.5.8* for detailed description of the algorithm used to determine when to deploy the rover and so initiate this action.



A motor holder made from 3D printed ABS as mentioned in the previous section held the motor in place. This holder must be reliable to ensure that the motor and so the latch mechanism itself remains centrally aligned with the rest of the can. If the mechanism were to misalign somehow during the can's descent or otherwise then it is highly likely that the mechanism will not release the two halves of the module correctly and so the rover will not be deployed successfully, hence the detailed analysis shown in one of the previous subsections.

2.5.2.4 Electronics Integration

Careful collaboration between the systems and mechanical teams was important in designing the landing module. The systems team had to know the maximum dimensions available for components and wiring, and the mechanical team had to know the exact dimensions of all of the components to ensure they would fit.

An important feature of the mechanical design of the landing module is the ease of access to the various electronic features of the module. As a microSD card was used for data storage, it is important that this was accessible for rapid retrieval of said data. Hence, cut-outs in one of the outer shells were made to allow for this and for access to the serial connector on the main landing module PCB for firmware uploads and debugging.

The base plate is cut out of clear acrylic, allowing the rover's camera to observe the descent and record images onto its SD card. This ensures that should there be an unforeseen fault in the mechanical system, the rover will have been able to complete part of its mission objectives. The top plate was designed to prevent the rover falling out of the module, and to connect with the parachute. This was achieved through the use of glue, due to the limited space available to incorporate a reliable solution that does not require the use of glue. The top plate will also be home to several small microswitches which will feedback to the system what state the latch release mechanism is in currently and when the motor should start and stop rotating to release the rover, for example.

2.5.3 Parachute development

To calculate the optimal parachute diameter for the terminal speed of 11 m/s the following formula was used:

$$v(t) = \sqrt{\frac{2mg}{\rho A C_d}} \tanh\left(t\sqrt{\frac{g\rho C_d A}{2m}}\right).$$

This is the function of velocity against time. Since the maximum value of tanh(x) is 1, the terminal velocity is given by:

$$v(t) = \sqrt{\frac{2mg}{\rho A C_d}}$$

Rearranging the formula to give the diameter as the unknown, we obtain:

$$D = \sqrt{\frac{8mg}{\pi \rho C_d v^2}}$$

Where: m = mass = 0.370 kg; g = free-fall acceleration = 9.81 m/s2; *p* = density of air = 1.2041 kg/m3; Cd = drag coefficient = 0.42; v = speed of CanSat = 11 m/s.

Resolving for D, we obtain D = 0.3886 m = **38.86 cm**.

To manufacture a parachute that would ultimately dome when subjected to up-thrust, it needed to consist of multiple segments, each designed separately in form of a petal, as opposed to a simple circular design. To do so, we laser cut templates so that we could accurately cut out (using a craft knife to minimise risk of tear, and to create a more accurate cut) each of the segments precisely into the required shape. The material used for the segments was a durable and lightweight parasheet, and each of the pieces was later sewn together. The parachute was then sewn to six pieces of kite string where the pieces met, which has a high tensile strength as well as being light, owing to its tightly braided structure.

2.5.4 Parachute Testing

We conducted multiple parachute tests, using both our segmented design and a simpler circular, one-pieced circular approach for comparison. Our drop tests were all conducted from an approximate height of 12 metres and were filmed with cameras to be able to analyse the descent of a mass, analogous to the final can in shape and mass, in slow-motion. We inspected a few criteria, such as ease and speed of deployment (to ensure the reliability of the parachute), the terminal velocity of the can and how far it strayed off course. Many repeats of this were conducted and some were included on our blog and YouTube channel as an opportunity for outreach. The experiment allowed us to safely conclude that the segmented parachute was more successful in all three target areas.

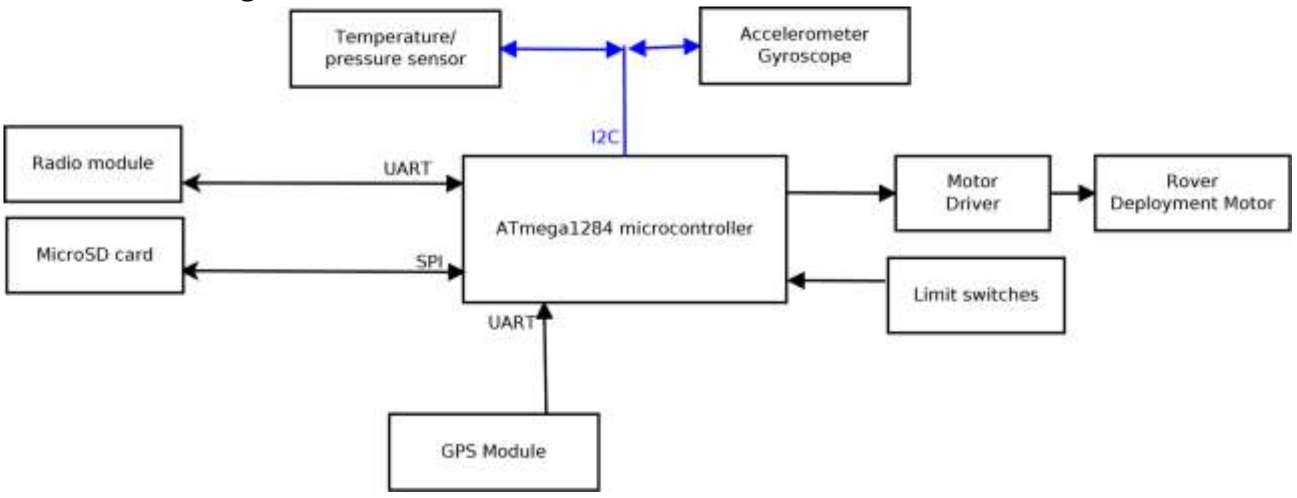
2.5.5 Landing Module Protection

The landing module itself is made of 3D printed ABS parts, designed to be flexible to distribute the force upon on impact, thus mitigating the damages caused. We decided not to use additional shock absorbing material at the bottom of the module because this would have been too bulky.

The more important part of the landing, however, is the successful decent, release and operation of the rover itself. Thus, we have incorporated a shock-absorbing system in the internal structure, whose primary purpose is to protect the rover and the integrated electronics from, for example, coming loose.

2.5.6 Landing Module Electronics

2.5.6.1 Block Diagram



2.5.6.2 ATmega Microcontroller

The core of the landing module electronics is an ATmega1284 microcontroller running the Arduino bootloader. This microcontroller was chosen because it has two serial ports, allowing one serial port to be used for software uploading and debugging, and one for the radio module. It also has I^2C for connection to the various sensors, and SPI for the microSD card.

A microSD card (chosen over SD because space is very limited) was used for data storage rather an EEPROM IC because there will be more data to store - the landing module should be able to log acceleration, temperature, pressure and location for several hours. EEPROMs would not be available in this capacity, and serial flash devices are often difficult to use.

2.5.6.3 Sensors

Due to manufacturing difficulties, some changes were made to the sensor configuration. Initially, we planned to mount the MPL3115A2 temperature/pressure sensor, MPU6050 accelerometer and Venus638 GPS module.

We instead decided to attach pre-made modules on flying wires to the landing module PCB. One contains a MS5611 temperature/pressure sensor[10] and MPU6050 accelerometer[11], the other is a GY-NEO6MV2 GPS module, as is used onboard the rover.

The temperature/pressure sensor, as well as being used as an altimeter to determine when to deploy, will also be used as a backup for our primary mission in case of a rover failure, allowing the landing module to relay back temperature and pressure, as well as store it onboard the microSD card. We also expected it to provide more accurate data than the sensor inside the rover, as the rover was likely to heat up internally from the power dissipated by the Gumstix, camera and motors, whereas comparatively little power would have been being dissipated inside the landing module, and there was also ventilation at the top and bottom.

[10] See: http://www.meas-spec.com/product/pressure/MS5611-01BA03.aspx [accessed 10/10/2013]
[11] See: http://www.invensense.com/mems/gyro/mpu6050.html [accessed 10/10/2013]

2.5.6.4 Radio Link

We have decided to use the radio module supplied in the CanSat kit for the landing module. Unlike the rover, where WiFi is required due to the high bandwidth requirements of video, the landing module only has to transmit position information and receive basic commands. As a result, this 433MHz module is more than sufficient.

The landing module transmits position, and atmospheric, data over the radio link. It also waits for commands, for example there will be the option of deploying the rover manually if the automatic deployment algorithm fails. The transmission of GPS data over the radio link aids recovery of our CanSat if line-of-sight is broken or otherwise obscured.

2.5.6.5 Motors

The landing module electronics had to support one motor for releasing the rover. There was also a microswitch as an end stop, to prevent motor damage from occurring. We used the 40 RPM motor described in *section 2.3.4.3.3*, as it had a high torque and is very efficient.

For the motor driver, we used the same driver as used on the rover - the DRV8833 - as the required specifications are very similar, and we had already thoroughly tested this part.

2.5.6.6 Power

We powered the landing module electronics off a single flat 3.7V lithium-polymer battery. Because the available capacity is relatively low, we had to make sure to design the electronics to run on the lowest possible power. We evaluated a range of batteries, and decided to use Nokia BL-4B batteries, as they are small, flat and have sufficient capacity for our application. They are also readily available from a number of sources. The motors are powered directly from the nominal 3.7V (range of 3.4-4.2V) produced from the batteries, as the motors still have sufficient torque and speed at this voltage, and the motor drivers can be run down to 2.7V.

Two additional voltage rails were required for the landing module electronics - 3.3V and 5V. The 5V rail is used for the ATmega microcontroller and CanSat supplied radio module. While the microcontroller can run at 3.3V, the maximum clock speed is limited to 8MHz at this voltage which could have been an issue, so we decided to use the 5V rail for optimum performance and reliability. The 3.3V rail is used for the micro SD card, GPS module and sensors. The 5V rail is driven by a boost converter (the TPS61072), as 5V is above the 3.7V input voltage. The 3.3V rail is driven with a LDO (low drop-out) regulator (the ADP124ARHZ-3.3) which should allow the battery voltage to drop to 3.43V before dropping out, as it has a typical dropout voltage of 130mV at a current of 500mA (the likely peak current with all devices running at maximum power). This was not an issue as the majority of the battery's capacity will have been used before the voltage drops below this point.

Measurements show that there will be a peak current consumption (motors running) of up to 1A (about 4W), and an average running current draw of up to 150mA (about 0.6W). This should provide a battery life of over 5 hours.

To comply with regulations, we included a power switch accessible from the outside that completely isolates power from the landing module electronics (this is wired in series with the batteries.)

There is a potential divider connected to the microcontroller which allows it to read the battery voltage.

2.5.7 Landing Module Software

The software running on the landing module will be kept as simple as possible. It logs data from the sensors to the microSD card, detect when the can has landed and deploy the rover, and handle the radio link. Data is stored on the microSD card in a CSV file, and a new file is created for each logging session. The following data was be logged:

- Time since startup
- Maximum accelerometer readings in that time period
- GPS position data
- Temperature and pressure data

Data will be both logged and transmitted over the radio interface at regular intervals, to give us a good set of data which will allow us to analyse the flight of our CanSat in detail.

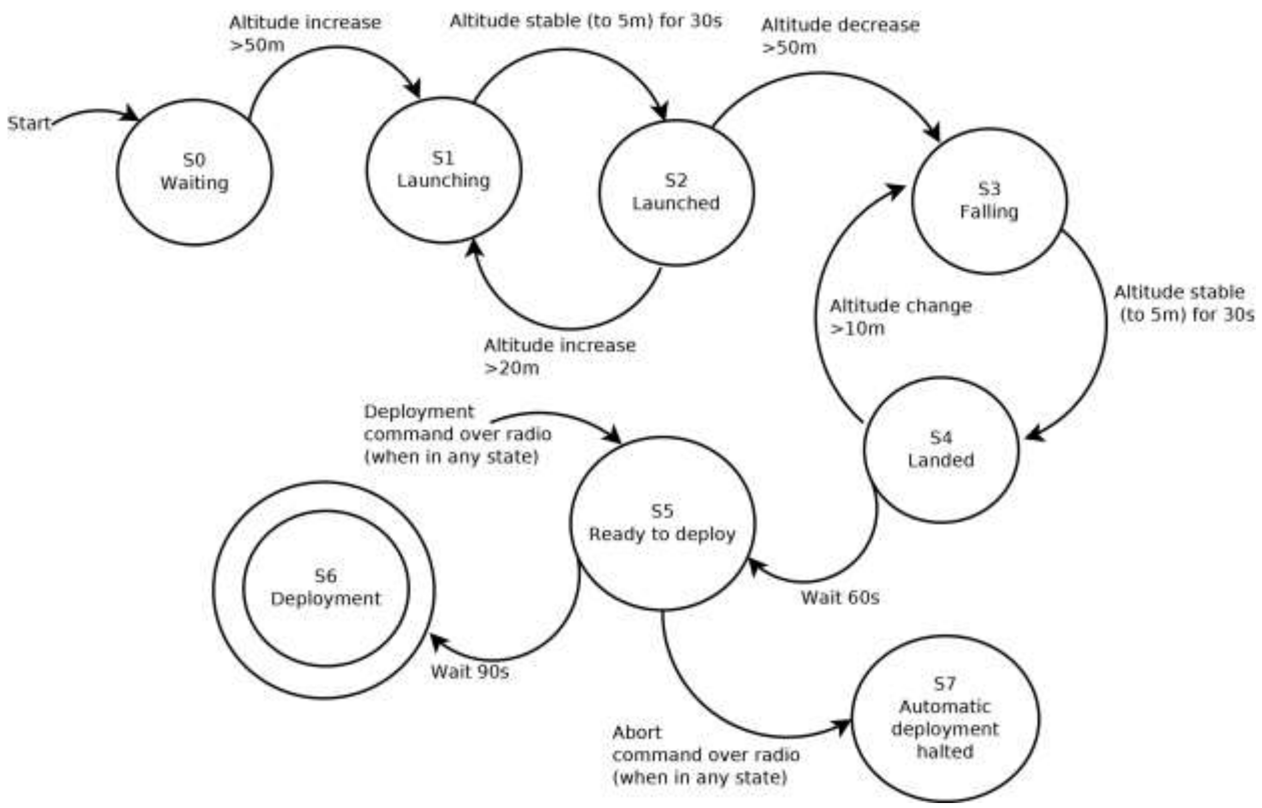
We used the library sdfatlib (http://code.google.com/p/sdfatlib/) to access and manipulate files on the microSD card. We also used third party libraries to access the various sensors onboard the landing module.

The use of a microSD card, combined with a standard FAT filesystem and CSV file format, meant that data could be recovered and graphed without the requirement of any additional software or functioning hardware, which would have been useful in the event the hardware is damaged on or after landing.

This software was written, examined and tested very carefully, as it is critical to the success of our project. If it were to release the rover too early, the rover would fall at terminal velocity and be destroyed. Likewise, if it fails to deploy the rover, then we will not be able to perform our secondary mission.

As part of the landing module software, we designed an algorithm that could accurately detect once the module has landed. We relied on altitude readings from the barometric pressure sensor.

The algorithm for determining when to deploy the rover uses a finite state machine. A diagram for this FSM is here:



Many stages are required in order to result in an automatic deployment, to prevent false positives accidently deploying the rover. In addition, a 90-second warning will be given over radio before automatic deployment, with the ability to send a command over radio to abort deployment if it is not safe to do so (i.e. it has not yet landed).

There is another purpose to the 90-second wait - as the CanSat is required to fall in 90s, even if deployment starts while the CanSat is still in the air, it will have landed before it actually deploys.

2.6 Ground Support Equipment

2.6.1 Hardware

2.6.1.1 WiFi (802.11) Communications

The data link between our rover and the base station runs on 2.4 GHz WiFi. As WiFi is not normally designed for distances of up to 1km, we decided to use a high gain directional antenna and high power router at the base station, to compensate for the lower power and gain on the rover.

The antenna is a 24dbi TP-link antenna, which was assembled . We did carry out some testing - though not across the full 1km we hoped to achieve. A picture of the antenna, taken during testing, is below:

This antenna is highly directional, having a field of view both horizontally and vertically of approximately 10°. Therefore, it was necessary to have some automated or semi-automated system to position the antenna. The initial position, used to establish initial wireless communications, is obtained from the 433MHz RF link to the landing module.

After initial connection has been made, the rover will streams back position data for tracking purposes.

Once this data is known, along with the position of the base station, trigonometry is used to find the horizontal and vertical angles that the antenna should be positioned to. If connection is lost (and therefore the base station software stops receiving position information), the software keeps the angles constant, as the rover's software always tries to take the most direct route possible back to the base station, therefore will remain in line.

As well as the antenna, we also needed a wireless router/access point. For this role we chose a high-power TP-link router, the TL-WA5110G, as this outputs at the highest power level legal in the UK, therefore giving the best possible chance of successful long distance communications.

2.6.1.2 Accurate Direction of Antenna

Initially, we considered using high torque servos; however, we could not find a design which could lift the heavy WiFi antenna. We instead changed to a system where the antenna is lifted and panned manually (with counterweights to keep it balanced) and the direction in which the antenna should be moved is displayed on a small LCD connected to the Arduino over SPI.

The LCD is controlled using an Arduino Mega connected to the computer via USB. The Arduino runs a simple program to receive target positions through USB (over the USB-serial interface) and displays guidance on the LCD. The antennas current position is determined by a digital compass and accelerometer (for determining the degree by which the antenna is tilted.) The accelerometer allows us to tilt compensate the digital compass.

We chose an LCD over LEDs as it allows the display of raw compass data as well as direction arrows which aided testing, debugging and set-up.

The position of the antenna is calculated by the ground control software based on initial data from the landing module, and then data from the rover once connected. In order to calculate the direction needed, we the software must first be configured with the position of the base station. The GPS on a smartphone can be used to obtain the coordinates of the base station.

2.6.1.3 433MHz RF Communications

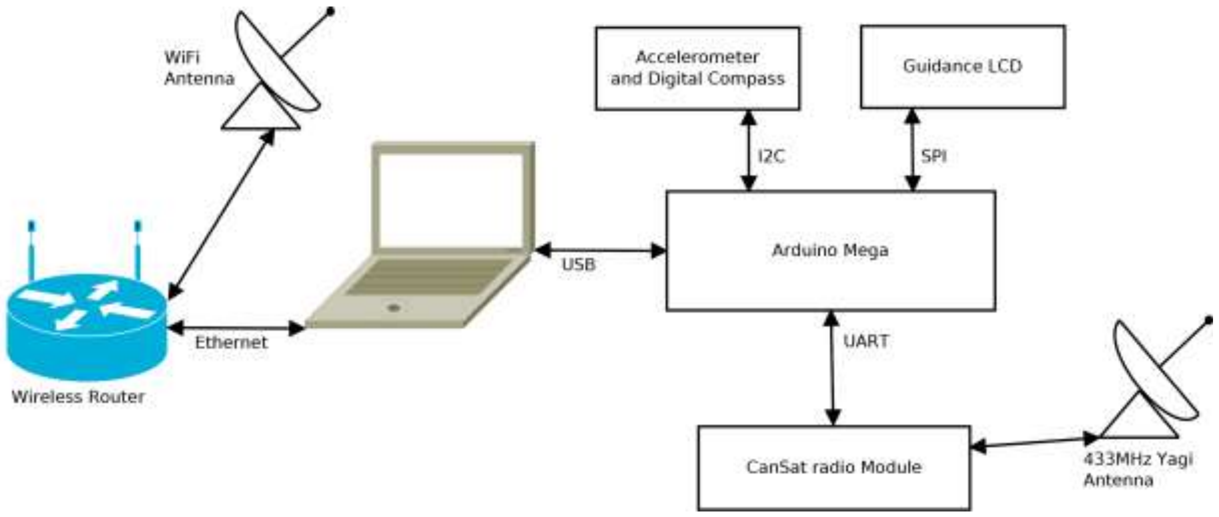
As well as the 802.11 link to the rover, there is a 433MHz link to the landing module, using the radio modules supplied with the CanSat kit. This streamed back an additional set of temperature and pressure, as well as location, data. Its purpose is to provide an initial position to point the wireless antenna at, and facilitate easy location and recovery of the landing module. In addition, there is the possibility of the computer sending a command to release the rover, in case a successful landing is not automatically detected.

The Arduino Mega also provides a USB interface to the radio module, using one of the serial ports on board the Mega.

We purchased a high gain Yagi antenna for these communications, in order to ensure the best possible chances of a successful connection. This antenna is also not very directional, which is important as this antenna is positioned by eye. As it is relatively light, and was only intended to be used for a relatively short part of the mission, we did not build a substantial mount for this antenna.

2.6.1.4 System Diagram

An overall block diagram of the ground station is below.



The wireless router is powered off a set of 8 AA batteries, and the Arduino and peripherals are powered from the laptop's USB port.

All electronics are enclosed in housings to provide a degree of protection, preventing the risk of damage by handling or light rain. These housings were made out of laser cut acrylic as this enabled us to have custom openings and markings where required.

2.6.2 Ground Control Software

2.6.2.1 Core Code

Whilst pooling together expertise in our programming knowledge, we decided that Java was the programming language to use, due to its ease of portability between operating systems. A basic GUI structure was built on JavaX.Swing, which proved versatile and flexible to use. The majority of the code was separated into different classes per component for ease of editing/collaborating. All collaboration between programmers was done by the Git source code management system with the aid of forks/branches.

To prevent faulty code from being pushed into the repository, an automated build of the Java software was set to occur for every commit. If the commit did not compile correctly, the person who authored it received an email. This functionality also allowed the whole team to download the latest version of the software from a central location. A small utility was also written that checked, on program startup, whether a new version of the program, or mapping data, was available, and downloaded it if it was. This ensured that everyone was running the latest version of the software.

2.6.2.2 Application Programming Interfaces (APIs)

Because it would be an unnecessarily large task and effort to write one's own API for the program, we sourced open-sourced APIs to use within the program. For the mapping API, we settled with OpenStreetMaps, as our first choice, Google Maps, did not support full offline mapping. The OpenStreetMaps allowed a pre-downloaded map, and flexibility. OpenStreetMaps is provided by the library JMapViewer. This library also made it easier to include real-time rover position updates.

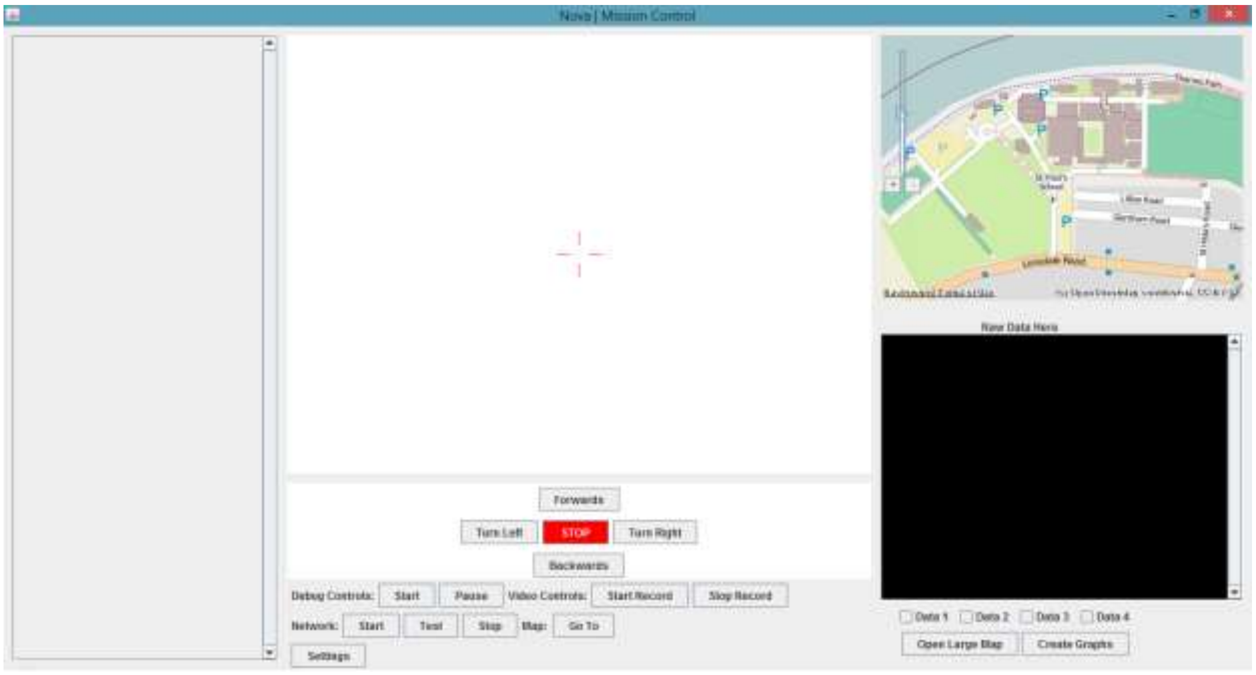
2.6.2.3 Protocols

Two protocols were used. A custom protocol based on raw TCP/IP sockets was used for control, and HTTP was used for serving images as described below.

For video, due to the limited broadcasting and transmitting options, the size has to be kept to a minimal. As processing power on board the Gumstix is limited, video was implemented as an image that updates several times a second, rather than a continuous 30fps video stream. This also makes the project a more realistic simulation of a real rover.

2.6.2.4 User Interface

We were initially faced with a number of decisions concerning the design of our user interface. This included the decision whether to have one window with everything, or multiple windows. In the end, we decided to go with one main window, but certain functionality can open as additional windows - for example a larger map.



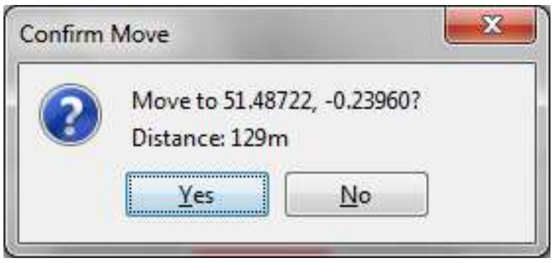
There are a number of parts to our user interface. The pane to the left displays debugging and status messages, for example when a successful landing is detected or when the rover reaches its target position.

The central white pane displays the video feed from the rover.

Below this, there is a panel containing five buttons allowing the rover to be moved manually, and to stop automatic navigation. There are also some controls that allow video recording, and pausing of the debugging console.

There is a display of the raw data received from the rover. This is the green and black text box in the bottom right corner. We had originally planned to include functionality for generating graphs from the received data, but this was never implemented.

In the top right, there is a map, implemented using JMapViewer, as described in *section 2.6.2.2*. As you can see on the picture, a red dot indicates the target position the rover is moving to, and the green dot indicates the current position of the rover. Clicking on the map allows the position to be set, but to prevent accidents the following message box is shown first:

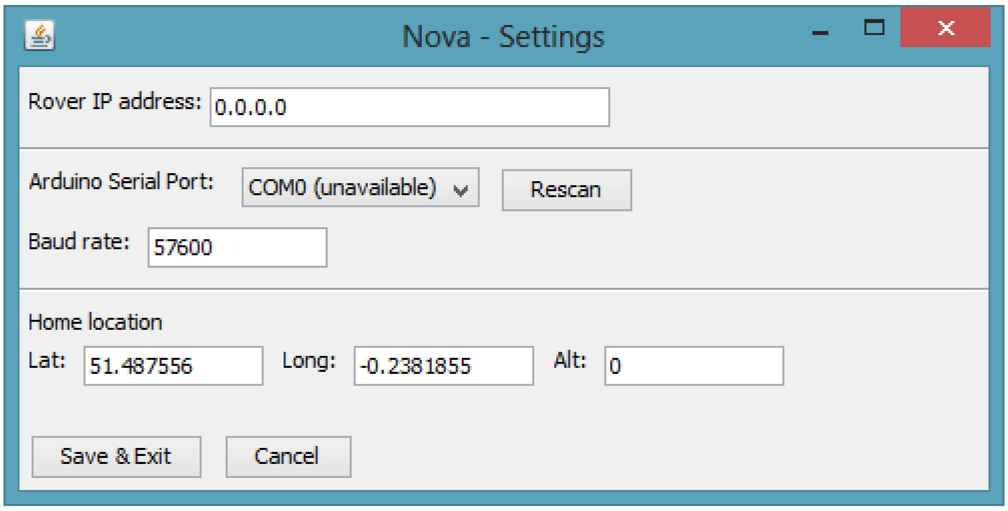

Confirm Move

Move to 51.48722, -0.23960?
Distance: 129m

Yes No

This displays the longitude and latitude the rover will move to, and the distance from its current position. After the user confirms, the target position is changed, and an update sent to the rover.

There is also a settings window, for configuring various options that may change depending on the exact usage configuration:



Rover IP address: 0.0.0.0

Arduino Serial Port: COM0 (unavailable)

Baud rate: 57600

Home location

Lat: 51.487556 Long: -0.2381855 Alt: 0

The available settings are:

- The IP address of the rover, that the application attempts to connect to
- The serial port that the base station Arduino is mapped to, and the baud rate to use for communications on this port
- A home location, which will be the exact location of the base station, used to set the initial display position of the maps and to calculate angles for antenna positioning.

2.6.2.5 Implementation Detail

Wherever possible, we took an object-oriented approach to programming so that we ended up with a program that is extendable, clean and easy to troubleshoot. This meant using classes and interfaces for many parts of these programs.

For both the rover and landing module, we defined interfaces called 'RoverUpdateListener' and 'LandingModuleListener'. Classes can implement these interfaces, and register themselves with the NetworkRover or LandingModule class and receive updates of the rover and landing module position and status. This means that adding new functionality, for example a live data view or logging to a file, is very easy.

Likewise, as we thought we may need more than one map, all of the mapping functionality (including our custom position markers) is contained in one, easy to use, class, called CustomMap, which extends the mapping library that we are using. The same is true of the class that provides the 'Go To' dialog, allowing users to select where they want the center of the map to be.

As video is provided as a series of images, described above, this means that an external library is no longer needed to display it. Instead, a function is called on a timer inside the 'NetworkImageViewer' class which fetches the most recent image via HTTP and draws it to the component.

2.6.3 Antenna Mount Design

The WiFi antenna was manually operated and it needed to be able to track the rover over a variety of directions and heights. Therefore there had to have at least 180-degree freedom in the horizontal direction and 30 degrees in the vertical direction. To do this we decided to mount the antenna on a rotational bearing joint (referred to as “Lazy Susan”), with the antenna stand attached to one end and a 50 cm by 50 cm base on the bottom.

The WiFi antenna was screwed loosely to the stand so it is possible to point it upwards and downwards, controlling the angle with a metal bar, which also serves as a counter balance. As described in *section 2.6.1.2* the operator uses a guidance system, so the antenna can be pointed in the relative direction of the rover.

3 Project Management

3.1 Planning

3.1.1 Tasks List

The task list below was drawn up at the start of the project in order to guide our scheduling and so we could monitor our progress. Each set of related tasks is defined within a single work packet (WP) for convenience.

	Task
WP1	**Systems Development**
	a Design rover electronics
	b Assemble rover electronics
	c Test rover electronics
	d Develop rover Arduino software
	e Landing module electronics design
	f Landing module electronics assembly
	g Landing module software development
WP2	**Rover hardware development**
	a Can size analysis
	b Motor choice and drive system
	c Measure up contents
	d 3D design iteration 1 and refining
	e Construction of iteration 1 and electronics integration
	f Final 3D design
	g Construction of final design and electronics integration
	h Function testing and troubleshooting
WP3	**Recovery system development**
	a Landing module size analysis
	b Technical specification
	c Mechanism design
	d Parachute research and design
	e Parachute manufacture
	f Landing module structural design 1, inc 3D
	g Design 1 construction
	h Integration of electronics
	i Testing and troubleshooting
WP4	**Communications Testing**
	a Test wireless link
	b Get video stream working
	c Test link to landing module
WP5	**Ground Station Development**
	a Ground control software development
	b Antenna mount construction
	c Project integration and software testing

WP6	**Outreach**
	a Complete website design
	b Launch website and social media
	c Societies fair/talks at school
	d Get an article in a local newspaper
	e Outreach with other schools
	f Dissemination of designs and results
WP7	**Competition tasks**
	a Prepare and submit progress report 1
	b Prepare and submit progress report 2
	c Prepare and submit final progress report
	d Travel preparation
	e Presentation preparation
	f Presentation rehearsal
	g Final project checks

3.1.2 Gantt Chart

We used a Gantt chart (based off the tasks list above) to plan the project. We aimed to allow some spare time (not shaded separately) for each task, and to leave ourselves leeway to make adjustments and fix things if necessary; retrospectively, we should have allowed more time still.

The team was relatively small, so a dependency table was unnecessary. Likewise, we elected not to use critical path analysis – each subteam aimed to complete their tasks as early as possible, and the complexity of the project meant that numerous tasks needed to be carried out simultaneously.

	Week Commencing									Half-term								Christmas			Month		
	19/08/2013	26/08/2013	02/09/2013	09/09/2013	16/09/2013	23/09/2013	30/09/2013	07/10/2013	14/10/2013	21/10/2013	28/10/2013	04/11/2013	11/11/2013	18/11/2013	25/11/2013	02/12/2013	09/12/2013	16/12/2013	23/12/2013	30/12/2013	Janurary	Feburary	March
WP1 – systems development																							
a Design rover electronics	🟩	🟩	🟩	🟩																			
b Assemble rover electronics				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩												
c Test rover electronics							🟩	🟩	🟩	🟩	🟩	🟩	🟩										
d Develop rover Arduino software								🟩	🟩	🟩	🟩	🟩											
e Landing module electronics design												🟩	🟩	🟩									
f Landing module electronics assembly														🟩	🟩	🟩	🟩						
g Landing module software development																	🟩	🟩	🟩				
WP2 - rover hardware development																							
a Can size analysis	🟩	🟩																					
b Motor choice and drive system						🟩	🟩																
c Measure up contents							🟩																
d 3D design iteration 1 and refining			🟩					🟩	🟩	🟩	🟩	🟩	🟩	🟩									
e Construction of iteration 1 and electronics integration															🟩	🟩	🟩						
f Final 3D design																		🟩	🟩	🟩			
g Construction of final design and electronics integration																					🟩		
h Function testing and troubleshooting																					🟩	🟩	🟩
WP3 – Recovery system development																							
a Landing module size analysis											🟩	🟩	🟩										
b Technical specification											🟩	🟩	🟩										
c Mechanism design													🟩	🟩	🟩	🟩							
d Parachute research and design										🟩	🟩												
e Parachute manufacture												🟩	🟩	🟩									
f Landing module structural design 1, inc 3D																🟩	🟩	🟩	🟩	🟩			
g Design 1 construction																					🟩		
h Integration of electronics																					🟩	🟩	
i Testing and troubleshooting																						🟩	
WP4 – Communications testing																							
a Test wireless link																🟩	🟩						
b Get video stream working																		🟩	🟩				
c Test link to landing pod																				🟩	🟩		
WP5 – Ground station development																							
a Ground control software development												🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
b Antenna mount construction														🟩	🟩	🟩	🟩						
c Project integraton and software testing																		🟩	🟩	🟩	🟩	🟩	
WP6 – Outreach																							
a Complete Website Design	🟩	🟩	🟩	🟩	🟩																		
b Launch Website and Social media							🟩																
c Societies fair/talks at school				🟩		🟩																	
d Get an article in a local newspaper																					🟩		
e Outreach with other schools																					🟩	🟩	
f Dissemination of designs and results																					🟩	🟩	🟩
WP7 – Competition tasks																							
a Prepare and submit progress report 1								🟩	🟩	🟩													
b Prepare and submit progress report 2													🟩	🟩	🟩								
c Prepare and submit final progress report																						🟩	
d Travel preparations																						🟩	
e Presenation preparation																							🟩
f Presentation rehearsal																							🟩
g Final project checks																						🟩	🟩

3.2 Team Management

3.2.1.1 Meetings

The entire team met weekly during Monday lunch breaks in a dedicated location at school. This meeting was often treated as a time where all the sub-teams could relay their progress to the rest of the team and the team leader. In this way, each sub-team was able to work relatively independently from each other, only needing to update the others on a weekly basis. This prevented team meetings from becoming cumbersome and unproductive.

During the week, the sub-teams met on a very frequent basis to continue work on the task they had been assigned to complete for the week. Sub-teams communicated within each other, only contacting the team leader when there was an issue or would benefit from more assistance.

3.2.1.2 Communication

A number of remote communication protocols were used for when the team needed to interact without being able to meet face-to-face.

3.2.1.2.1 Google drive

Google Drive was used for storing documents for collaborative editing, as well as a small number of static documents (such as the competition guidelines). For example, we maintained documents containing records of meetings, sizes, costings and the location of parts and tools. We also made sure to update our ‘ideas’ documents following discussions in which creative suggestions and choices were made.

3.2.1.2.2 Dropbox

Dropbox was used for file storage, particularly for files of a non-standard format (such as 3D CAD files).

3.2.1.2.3 Email

Remote written communications were generally conducted by email (rather than instant messaging). Emails are more ‘permanent’ allowing us to easily refer back to previous topics of discussion. In addition, certain team members preferred not to use other online methods of communication (such as Facebook).

3.2.1.2.4 Skype

Skype was used for detailed discussions using VOIP. Skype proved ideal for times when we were unable to meet at school (such as during the holidays) and also when unforeseen problems turned up that necessitated technical discussions of more detail than would be possible over email.

3.2.1.2.5 Github Repository

The Systems team used Github to collaborate on the code used in the project. Automatic checks were implemented to prevent faulty code from being uploaded, and on some components of the project, like the ground station, utilities checked the repository for an updated version on program startup. This ensured that everyone working on or with the code had the latest versions.

3.3 Resources

3.3.1 Physical Resources Used

3.3.1.1 Machinery

Throughout the design and prototyping phase, we used a RepRapPro Huxley 3D printer to create physical prototypes of our designs. 3D printed models allowed us to visualise how much space we would have in the final design, as well as allowing us to work with a physical prototype and build structures around it. The advantage of using a RepRapPro 3D printer was the minimal material cost: the cost of printing 1cm^3 of material on the RepRap was significantly lower than that of the HP DesignJet 3D printer, owned by the school. In addition, the RapRap was kept at Nicolas Weninger's home, which means that work could continue over the weekends. The HP DesignJet was used for the final model due to its greater accuracy and use of support material.

3.3.1.2 CanSat Hardware

During the design phase of the project, we needed to order several different types of the same component, in order to test them for suitability. For example, several different DC motors were ordered. The costs of these were all logged in the costs and accounting spreadsheet. We also often bought spares of components in case we wanted to use some for testing, or some were damaged during production or testing. The school itself also had many of the resources we have used; however, anything that they did not have, we were able to order.

3.3.1.3 Electronics Assembly

The soldering equipment in the school electronics lab was deemed insufficient for the surface mount (SMD) work we needed to do. For this, a temperature controlled iron with a fine tip, and ideally hot air gun, was necessary. As a result, the surface mount soldering was done with a Yihua 898BD+, kept at David Shah's house. This also allowed work during holidays and weekends. Surface mount components were necessary for two reasons. Firstly, there was a large constraint on size, and surface mount components are significantly smaller, and secondly the connectors used to connect the Gumstix to the mainboard had a very small pitch so were only available in SMD form.

Most of the assembly was done using a conventional temperature controlled soldering iron, leaded solder, solder wick and a flux pen, using 'drag soldering' techniques; however a hot air gun was used for the connectors to the Gumstix and MPL3115A2 sensors. This is because the Gumstix connectors were a very fine pitch, and the MPL3115A2 sensors had a leadless package so were impossible to hand solder.

3.3.2 Budget

Since the start of the project, the team kept regular logs of expenditure on a shared spreadsheet, detailing item cost, order cost and a link to the product online, amongst other fields. This was regularly updated whenever a new order was placed for a component, web hosting, PCBs or anything else associated with the project. Using this, we were able to accurately determine the cost of each item used in the can.

In terms of funding available, the school had a generous sum of money set aside for CanSat, obtained through the SPS Space Society. In addition to this, four team members have been awarded Arkwright engineering scholarships by the Arkwright foundation. The Scholarship grants each student £600 over two years, as well as £400 to the school per student. These funds were available to draw upon should we have exhausted the school's budget.

In addition, Gumstix and RepRapPro kindly sponsored us and provided certain materials and pieces of equipment free of charge.

3.3.3 Support and Guidance

3.3.3.1 Internal Support

The school has an extensive engineering and technology department with five dedicated staff members, who all specialise in different areas of engineering. Dr Patterson, the link teacher, advised the team based on his experience with previous CanSat teams. The Workshop staff were always on hand, to whom practical questions were directed. For example, if we needed advice regarding how a fixture might work and its suitability, the workshop staff were more than happy to assist.

The school Robotics Society - RoboSoc - as well as the Engineering Society - EnSoc - were also available for advice and resources.

3.3.3.2 External Support

RepRapPro and Gumstix, our two sponsors, offered to give us assistance in their respective areas should we have required it. As four team members are Arkwright Scholars, as mentioned above, the team had contacts in industry it can call upon should the situation so have demanded. However, there was little need to contact any of these people apart from Gumstix, for technical questions.

3.4 Health and Safety

3.4.1 Use of Machinery and Tools

Care was taken while using all machinery and tools, particularly power tools (such as pillar drills). All team members wore safety glasses while doing anything that could result in flying pieces of material (e.g. drilling, using the Dremel) and removed/secured any loose clothing while using machinery in which it could be trapped. Team members were also trained in the use of unfamiliar equipment by members of staff, and we made sure that they would not be distracted while the equipment was turned on.

Team members were careful to solder only in well ventilated areas, using safety glasses where appropriate.

3.4.2 Electronics

Although we were generally only working with low voltages and currents, care was still taken when working with the circuitry while live.

Lithium batteries (which have a small risk of catching fire, particularly during charging) were required for their large capacity. To reduce the risk of fire, the rover featured a latch to remove the batteries for charging (to prevent overheating); batteries were always charged on a heatproof mat.

3.4.3 Other

Large amounts of equipment were required for the competition, and some parts of the antenna mounting system were particularly heavy. Team members were careful to carry only as much as they felt comfortable with, and not to carry large loads for a long period of time.

As the motors used had large amounts of torque, care was required while handling the CanSat when powered up to avoid trapping fingers. To this end, team members were particularly vigilant while anyone was working on the CanSat while powered up.

Throughout the project, decisions were taken with health and safety in mind; we aimed to foster an attitude of vigilance and care within the team to reduce the potential for mishaps.

3.5 Test Procedures

This section outlines the test procedures we carried out prior to the competition. We aimed to be as thorough as possible with our tests, putting parts of the project through tests under many different conditions.

3.5.1 Wireless Communications

One of the more experimental sides of our project was long range WiFi communications. We carried out both short-range and long-range checks on the stability and range of the system.

We also tested the Yagi antenna that was used for communication with the landing module. This was tested primarily to ensure the system was working, rather than to test range; we believed that the power and gain of the antenna would more than meet our needs.

As well as checking communication could be made, we also checked that communication was reliable enough for control and video streaming.

3.5.2 Rover electronics

We thoroughly tested all parts of the rover electronics in all conditions. This included testing across the full range of expected battery voltages, to ensure reliability, as well as ensuring the system failed safely across common faults, for example stalled motors or partially disconnected subsystems. The reason for this was that these will likely occur during operation.

We then conducted functional testing to ensure that the rover performed everything that we listed in our functional specification correctly. We wrote a comprehensive set of test programs to ensure all of the electronics were functioning correctly and reliably.

3.5.3 Rover software

Closely linked to the rover electronics was the software that runs on board the rover. We tested the following functionality: network command handling, video streaming, connection control and navigation and motor control.

We also took particular care to ensure that the software remained functional even if the wireless network was lost and regained multiple times, because if the software were to crash in this case we would not be able to re-connect to the vehicle.

3.5.4 Ground station software

This was the most complicated part of the software, so testing was complicated. Like with the rover software, part of the testing ensured that all program paths were covered and that the functional specification was met. To check suitable behaviour, we used a packet analysis tool to ensure that data was sent correctly across the network.

Another part of testing this software was ensuring mapping was accurate. This was important as unreliable or inaccurate mapping would have made controlling and retrieving our CanSat very difficult.

3.5.5 Landing module

The electronics, mechanical parts and software on board the landing module were critical. If these failed, the rover could either fall out during descent, which would destroy the rover and onboard electronics, or jam and not release the rover, in which case we would be unable to fully carry out our secondary mission. Testing was thus critical. As well as testing that everything was functional under normal conditions, we also tested under partial fault conditions. This aimed to ensure that even in event of a partial or temporary fault, the rover would not be released early, and ideally could still be released when required.

3.5.6 Mechanical strength

As part of our test procedure, we tested how robust the design of our system is when falling at $11ms^{-1}$. Initial tests were done on separate mock-ups, to evaluate the effectiveness of different methods of padding and crumple zones. Similar protective procedures were then applied to our actual CanSat.

4 The Launch Campaign

4.1 Run-up Preparations

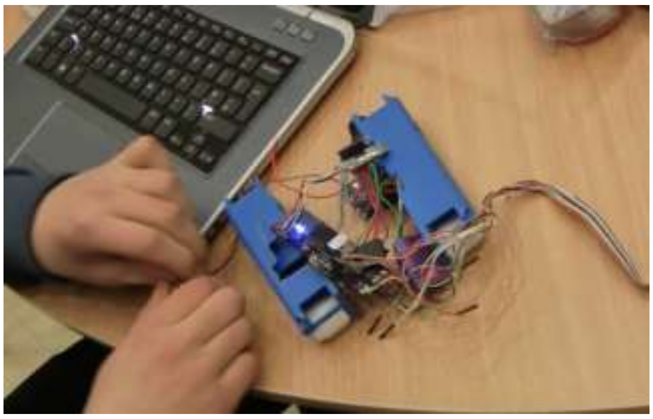
In the days running up to the launch campaign, items of the project were still not functional. Due to the issues we experienced with HP DesignJet 3D printer jamming for the previous three weeks, we did not have the opportunity to print a third revision of the design and did not have time to fully assemble the second revision either, as it was only printed with a few days to go to the campaign.

Nevertheless, the team remained after school until usually 10pm in the days before departing for York, the location of the launch. These sessions involved final integration of all aspects of the hardware and electronics, combined with testing of the entire system. The testing revealed some shortcomings in the system, as well as other issues most likely brought about by the integration process. For example, as the team had planned for a third revision of the rover chassis, some PCBs did not fit as well as hoped. This meant that the model had to be carefully modified using a Dremel tool.

After this was completed, we conducted an initial power-up. A short circuit was present between the camera module and the Gumstix, which resulted in the ribbon cable connecting the two becoming unusable. Unfortunately, as this was a proprietary cable, there was no way in which this issue could be resolved. We spent the remainder of the final evening removing bits of code that would have given errors with the absence of the camera and packing the crates of kit we would need on the trip.

4.2 Final Preparations and assembly

Following the train journey to York, we began by unpacking our kit at the National STEM centre at York University. While there, we continued to test the hardware and software. In doing so we stumbled across some more wiring errors, which were promptly corrected. In the evening, the team decided to attempt to find a replacement camera for the mission. We took a trip to the local Maplins – with the taxi driver making an accidental trip to Matalan instead – to source a standalone camera unit, which we planned to integrate into the rover. A "spy pen" provided a useful replacement, and using the tools we had brought with us from school, we modified the chassis in order to integrate this module. Unfortunately, the disassembly of this product resulted in some damage to the PCB of the camera and despite efforts to re-solder the dislodged component, the camera failed to power on.



On the first evening, the team worked on final integration of the project in preparation for the scheduled launch the next day. By 11pm, all systems were functional, and it was decided that the CanSat was ready for launch. We checked our checklists for the next day and went to bed.

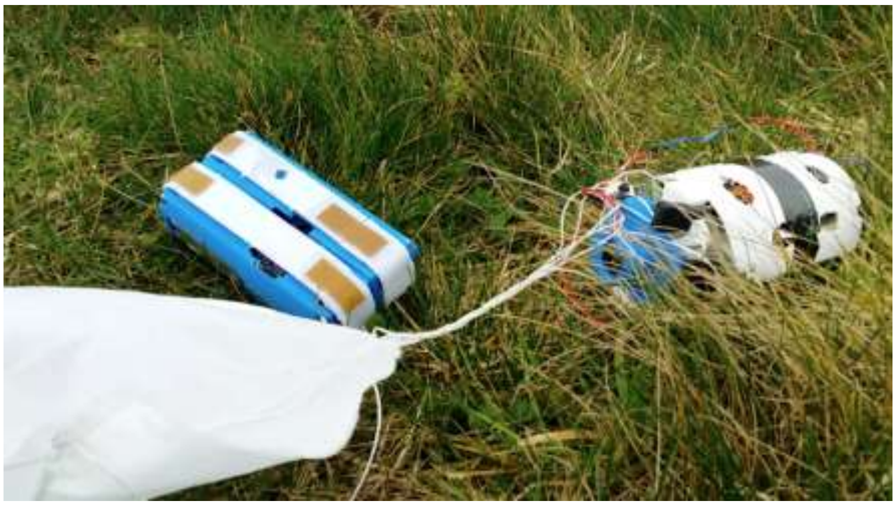
4.3 Pre-launch Presentation

The launch was postponed on the second day due to gale-force winds, prohibiting the safe operation of the launch vehicle by the competition staff. We disassembled our ground station and antenna mounts for transportation back to the STEM Centre from Elvington Airfield. On the way back, teams were instructed to prepare to present their project presentations to the panel of judges. Unfortunately, due to the fact that we had spent the previous nights preparing the CanSat for the competition, we had left the presentation largely unfinished and had hoped to complete it during the evening following the launch. Instead, we were given just over an hour to complete it. Combined with not having a reliable Wi-Fi signal, as the preparation area was in the basement, we struggled to collate the images we wanted in the slide show and distribute the recently written script to various computers so that the different team members could refer to it during the presentation.

4.4 The Launch

The launch took place on the next day following the postponement. We reassembled our ground station and conducted final checks for launch. Due to the fact that a third revision of the can could not be completed, we did not have the opportunity to refine and minimise the design. Thus when it came to launch vehicle integration, the CanSat was slightly too big for the launch vehicle. We came up with a resolution to this that involved a strip of duct tape and careful manoeuvring of the CanSat into the launch vehicle.

Upon landing, the landing module failed to deploy the rover. The motor responsible the unlatching mechanism was functional, but due to the landing module testing in the previous days, the mechanism had worn down. However, the rover was fully functional, but only after we manually released it from the landing module.



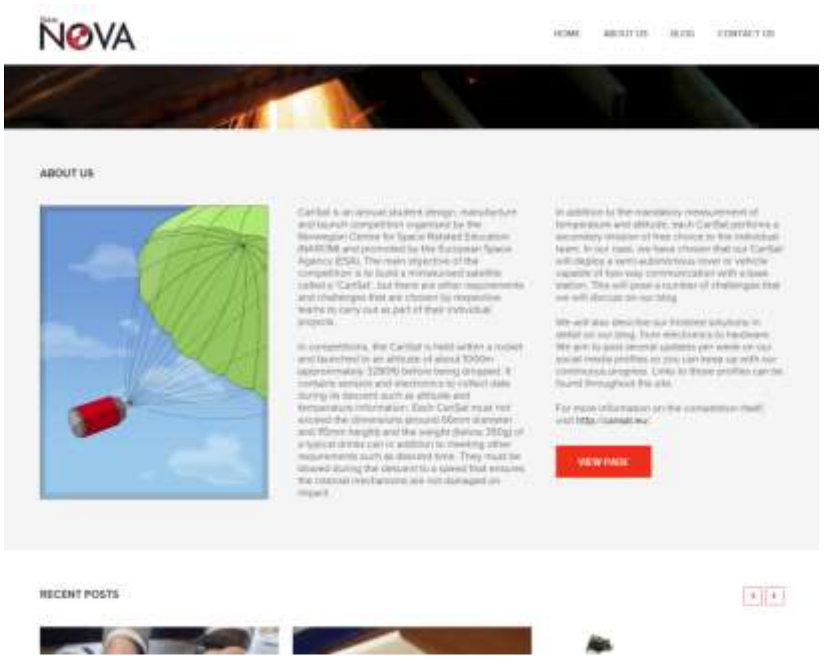
4.5 Post-Launch Presentation and Results

Returning to the STEM Centre, teams were given 45 minutes to extract the mission data from the CanSat and compile a post-launch presentation. Unfortunately, the temperature and pressure sensors in the landing module returned erratic data, suggesting that either a PCB trace or the sensor itself had broken during the ascent or descent. Likewise, the data from the rover sensors was not very useful, as neither the pressure nor temperature changed very much during the launch. This may be due to the fact that the CanSat only ascended approximately 70m before falling out of its makeshift enclosure. We did however present the judges a video of the functioning rover after impact.

At the awards ceremony, we were disappointed to hear that we were not awarded first place, but we were nevertheless pleased with what we had achieved.

5 Outreach

5.1 Website and Social Media



[illegible]

Josh Efiong, the outreach team leader, designed and constructed a site from scratch, which went live at http://www.teamnova.co.uk/. It is based on WordPress, the content management system. We decided to approach our website in this manner due to the freedom that this provides, both in the information we are able to present and in allowing us to implement our own defined list of features, over an off-the-shelf solution. We wrote regular, detailed updates on reaching major milestones in the team's progress since the start of the school year. These posts have covered all aspects of our progress, including our other outreach projects, electronics development, program specifications, hardware and so on.

Online presence

Website: **teamnova.co.uk**
Facebook: **facebook.com/cansatnova**
Twitter: **twitter.com/cansatnova**
Youtube: **youtube.com/user/cansatnova**

We actively encouraged users of the site to comment on posts and get in contact with us. We constructed a wide ranging social media portfolio that we feel helped us publicise our team and the competition to a wider audience. We had Twitter and Facebook pages on which we posted consistent and regular updates on what we achieved on a session-by-session basis. These tended to be brief messages about something we accomplished in the workshop, an aspect of the electronics that was worked on, some CAD that was completed or a talk that certain team members gave on a particular day. We also had a YouTube channel, onto which we posted videos of our testing in addition to other things. We developed a brand that was consistent across all our electronic profiles.

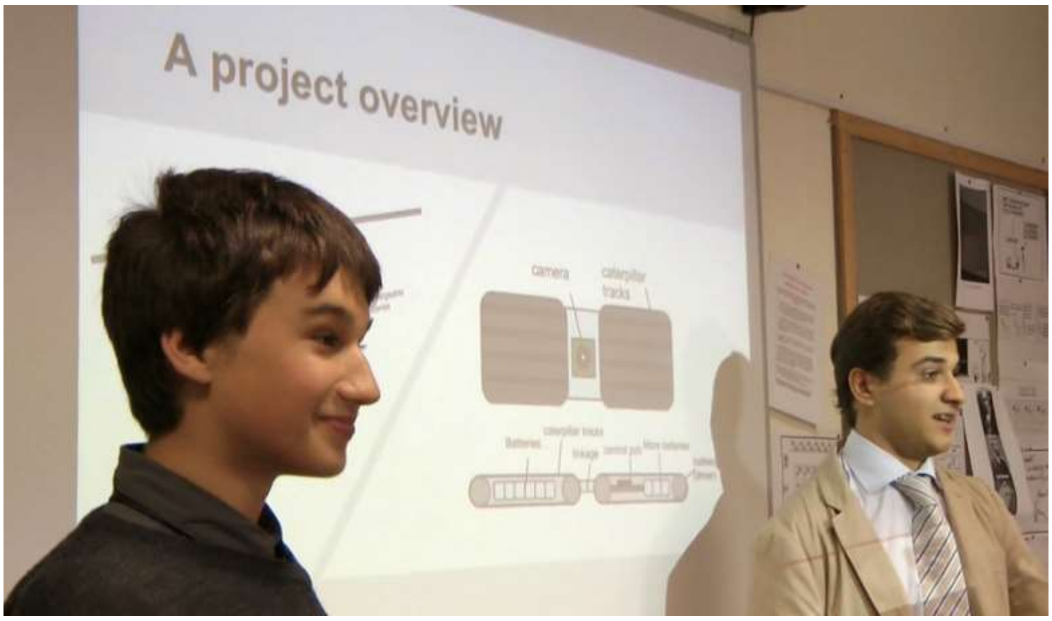
5.2 Talks to SPS Space and Societies Fair

SPS Space is the school's aerospace and aeronautical engineering society. Two team members as well as Dr Patterson - the link teacher for CanSat - were heavily involved with forming this new society. Since then, it became the backdrop of many of our outreach efforts.

Towards the start of the academic year, the school arranges the Societies Fair, whereby all societies in the school set up a stall and present their offerings to new and current students alike. SPS Space was given a prime location on the floor, allowing us to attract a sizeable number of interested students with a demonstration of many fascinating projects, not least the previous year's CanSat entry from St Paul's, one of whose team members was also there to assist us with the stall. A large number of students subscribed to receive regular updates from the society and the projects it is supporting, including CanSat 2013/14.

SPS
SPACE
VORTEX
BOYS TO PORTERS LODGE
Collecting directional information on radioactive particles at altitude

Following this event, David and Nicolas, the systems team leader and the hardware team leader respectively, gave a presentation to the society. This covered the projects we had previously undertaken, our CanSat entry and how members of the society can get started with their own projects. We then set them off in groups to brainstorm and organise interesting projects of their own.



5.3 School Magazines

In an effort to gain greater exposure around the school, we arranged to place articles in two student run magazines.

We wrote and submitted two articles to Black & White, the school's most widely read magazine. With a print run of approximately 500 copies twice every term, this enabled a wide audience to find out about CanSat and Team Nova. We hoped that readers followed up upon reading the article by finding out more about our progress using Facebook, Twitter and teamnova.co.uk.

We also arranged to place an article in Psci, the school's student run science magazine. This is distributed free of charge to all students studying science in the lower school.

5.4 Sharing Resources with the Online Community

One of our key outreach goals was to make as much of our project as possible available, open source, to the online community. We did this through our open Github repository at github.com/daveshah1/nova, as well as by encouraging people to contribute to the forums we ran on our website.

5.5 Launch Campaign Videos

Throughout the launch campaign, the team filmed themselves working and the final preparations before launch. These were edited together late in the night following the day's activities so that they could be uploaded for interested viewers to keep track of the team during the campaign. In total three videos were uploaded and can all be found on our YouTube channel.

5.6 Other Outreach

Two team members gave a presentation on the physics behind the telecommunications systems used by the CanSat to a Year 12 Physics class; this encouraged members of the class to take more of an interest in CanSat, and the applications of physics in aerospace engineering.

Furthermore, an international science conference is held every year in London attended predominantly by English and Japanese schools. We presented a poster at this event (which took place during the launch campaign!) and aimed to encourage visitors to the fair to follow our progress and consider aeronautics in a new light.

6 Reflections and Conclusions

6.1 Mission Results

6.1.1 Extent of Success in Primary Mission

Summary of objective	**Extent of success**	**Justification**
Temperature and pressure data are recorded and transmitted at a frequency of 1Hz	**Partially successful**	Although temperature and pressure was recorded by one set of sensors and transmitted back to the base station at an appropriate frequency, the accuracy of the data was questionable. The pressure data fluctuated in a way that would suggest some type of technical fault; we were unable to determine the cause of this error. The temperature sensor appeared to function correctly, but due to the design of the CanSat it was significantly affected by the heat given off by the Gumstix COM. However, the altitude from which the CanSat was dropped would not have resulted in a large temperature change; we suspect that if the temperature were to be allowed to stabilise, the temperature change while dropping from a higher altitude would have been recordable.

6.1.2 Extent of Success in Secondary Mission

Summary of objective	**Extent of success**	**Comments**
CanSat complies with size and weight specifications	**Partially successful**	The initial CanSat design weighed approximately 365g including all batteries and removable items. This meant it weight was very close to the specification weight of 370g, so it would not have been necessary to remove parts of the CanSat to reduce weight or carry ballast to increase weight. The specification weight was later revised to 400g shortly before the competition; small modifications to the CanSat meant its final weight was 400g exactly. Although the rover itself fitted well within the size limit, the landing module was unfortunately approximately 3mm wider than the specification width, and significantly longer. Although the additional length was not of great importance, the additional width meant the CanSat was unable to be launched in the standard manner.
Damage during landing is minimised	**Successful**	No parts sustained any damage as a result of the landing. The parachute slowed the CanSat to approximately 11m/s, the specification value, and the landing module successfully protected the rover (including the more delicate mechanical parts).

		The rover was unable to exit the landing module without assistance following the launch; however, this was due to a design flaw and was not caused by the impact.
Rover can traverse a wide variety of challenging terrains	**Highly successful**	The rover was extremely adept at reliable travel over many different terrains, including long grass. This was largely facilitated by the tracked design and the use of a high-torque drivetrain. The following surfaces were among those tested: • Hard earth • Long grass • Loose gravel • Carpet • Lino The rover was able to operate upside-down if necessary, and was able to surmount very significant inclines (although we did not specifically measure this feature).
Two way telecommunications, including a video stream	**Mostly successful**	Two way telecommunications were fully functioning at the time of the launch over a reasonable range. We were able to send commands to the rover, and were also able to receive data back. However, although the camera was completely operational during testing a flex cable was short-circuited during assembly; we were unable to source a replacement for this part, so we were unfortunately unable to record and transmit video footage.
Both semi-autonomous and manual control of rover	**Successful**	The rover responded correctly to commands over Wifi. A rudimentary semi-autonomous driving algorithm was also implemented.
Functioning and relatively complete base station interface	**Highly successful**	The base-station software was extremely fully featured. It worked reliably, and provided access to huge amounts information transmitted from the CanSat and landing module.

6.1.3 Summary of Mission Success

Overall, we were mostly successful in fulfilling our criteria for success.

There were some extremely impressive technical achievements, most notably the rover systems (including electronics design, assembly and programming), the multi-terrain capability, and the base-station software suite.

The two areas in which we made significant errors were temperature and pressure recording and the size of the CanSat. Both of these errors could have been remedied by constructing a further revision of the CanSat, with a slightly smaller chassis and landing module design as well as cooling vents for the Gumstix.

6.2 Analysis of the Project Outcomes

6.2.1 Mechanical Design

The mechanical design on the whole was well-considered, took into account a wide variety of potential operating environments and was completed punctually.

The drivetrain and tracked design performed extremely well. A number of tests were carried out in order to determine the best systems to use; these tests were extremely helpful in choosing what material to use and how to mount the motors.

The rover's shell was quite compact and was of a good shape. There were a large number of supporting structures inside the shell for segregating and protecting the electronics; although these were not necessarily in the wrong positions, there was no space left for ducting wires and assembly of the rover was extremely challenging and time consuming. Extensive modifications were required using a Dremel and other hand tools in order to make the supporting structures operable. It may have been better to leave them out completely; this approach would have slightly reduced the overall size of the rover too.

6.2.2 Electronic Design

The electronics designs were highly effective and were completed punctually, despite the extreme level of complexity. Necessary specifications, including size, were complied with and all desired functions/features were included. Component choices were appropriate for the context.

Although programming did take longer than expected, the resulting programs were functionally complete and contained numerous fail-safes and were designed to continue operating in the event of an error.

6.2.3 Manufacture of Rover

3D printing the structural and mechanical parts of the rover was a quick and effective method of manufacture, although it was quite expensive. In addition, when the 3D printer broke we were left without a feasible second construction method.

PCBs were manufactured professionally, which improved their quality (manufacture within school would not have been possible); soldering and assembly was conducted using appropriate machinery, and was completely successful.

Assembling the rover was challenging, but the use of nuts and bolts meant it was easy to disassemble as required. Using fewer internal support structures would have made assembly much easier (see above). The use of a Dremel speeded up the modifications that were necessary, although ideally these could have been predicted and included in the 3D designs before printing.

Overall, the rover was manufactured to a very high standard; the rover was finished in strong agreement with our designs.

6.2.4 Design and Manufacture of Landing Module and Recovery System

The necessity of the landing module was questionable – it may have been easier to attach the parachute directly to the rover. However, the landing module design we did select was one of a large number of potential solutions. The particular challenge was designing it such that the rover could exit no matter what angle it landed at – our solution would have had a highest chance of success had it been slightly larger (so the rover did not get stuck inside).

Aside from the internal dimensions, the design and construction of the landing module were relatively successful – the small amount of space available for electronics was used well, the mechanism contained as few moving parts as possible and the aluminium supporting structures were sufficiently strong to protect against impact and support the large tension forces during parachute deployment.

The parachute design balanced complexity with stability and descent rate satisfactorily. The design and assembly fulfilled all necessary criteria.

6.3 Analysis of Project Management

6.3.1 Planning

Our tasks list proved to be very comprehensive, covering almost all tasks we eventually needed to carry out. However, the timings illustrated on the Gantt chart were often extremely ambitious, particularly with regard to the amount of time taken up by programming. Were we to repeat the project, we would allow more time for programming and aim for slightly simpler results.

Our plans were thrown out by one month because our 3D printer jammed. We were unable to recover completely from this; although the rover was constructed by the deadline, our testing was seriously curtailed and we were unable to make important adjustments by producing a new version. Although allowing any more contingency would not have been possible, a slightly simpler project that either did not require so much complicated equipment or could be completed faster would have reduced the damaging effects of delays.

6.3.2 Communications

Team members reliably stuck to communications guidelines set out at the start of the year. We very rarely had any problems contacting people remotely, which was of great benefit to the progress of the project. Shared files were frequently updated and shared storage spaces remained an invaluable asset.

Although we would not necessarily change any of our communication protocols were we to begin a new project, we would consider using dedicated task management software; however, this would not necessarily be practical for small projects with small teams.

6.3.3 Resource Management

Parts were generally ordered promptly, and were generally not lost or misplaced. Likewise, we kept track of tools and machinery across a variety of locations.

The project cost more than we had planned for; this could have been averted by reducing the complexity of certain peripherals and by ordering fewer spare parts (although that could potentially by a dangerous strategy for key components).

6.3.4 Health and Safety

There were no recorded health and safety incidents which occurred as a result of working on or handling the CanSat.

6.3.5 Testing

Our testing of the final CanSat was curtailed by a broken machine, but testing of all separate subsystems was carried out as per the testing plans. Only one part of the project failed due to insufficient testing (but its failure was nevertheless rectifiable due to delay cause by the broken machine).

6.4 Success of Outreach

Our outreach took place over several different mediums; this meant we were able to reach a wide variety of different audiences, both inside and outside the school.

Within the school, we published two articles in the largest internal magazine (with a distribution of 500 people). We also held a talk to publicise engineering and CanSat and took part in the annual Societies Fair; both events attracted many students in the junior school. We also attended an international science conference, where we presented a poster.

Externally, we posted frequently on our blog and social media, and kept our public code repository on GitHub well updated.

Although on the whole our outreach was relatively successful, we unfortunately did not secure publicity in a local (or national) newspaper; this would represent a higher priority were we to commence another similar project or competition.

7 Conclusion

Throughout this project, team members worked extremely hard to create a successful device that would fare well in the national competition. Despite the few partially successful objectives listed above, the team is very happy with the outcome and are proud to have represented Team Nova at the national finals. We all learned a great deal through this experience, not only with respect to technical and engineering skills, but also team and project management skills. The team are very grateful to have been given the opportunity to take part in such a competition, and would highly encourage others to get involved in UK CanSat.

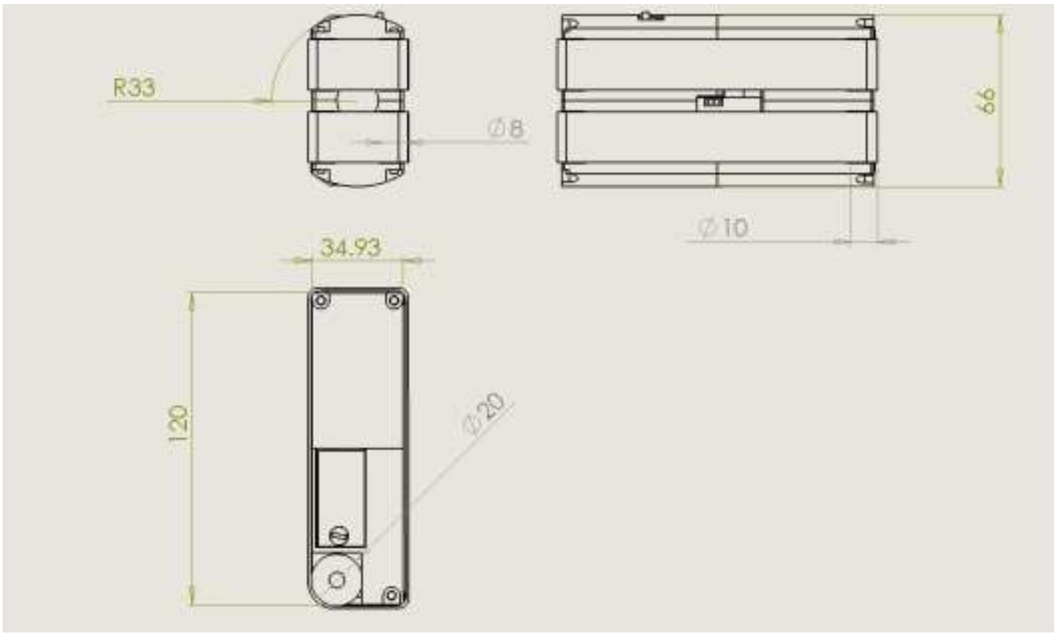
The team would like to thank St Paul's School and St Paul's School Engineering for offering their resources and to Dr Stephen Patterson and Dr Thomas Weller for their unrelenting support during the project and accompaniment to the launch campaign. Aditionally, the team would like to thank the sponsors, RepRapPro and Gumstix for their generosity and support. Many thanks also to Tom Lyons and The University of York for arranging the competition and hosting us during the launch campaign.

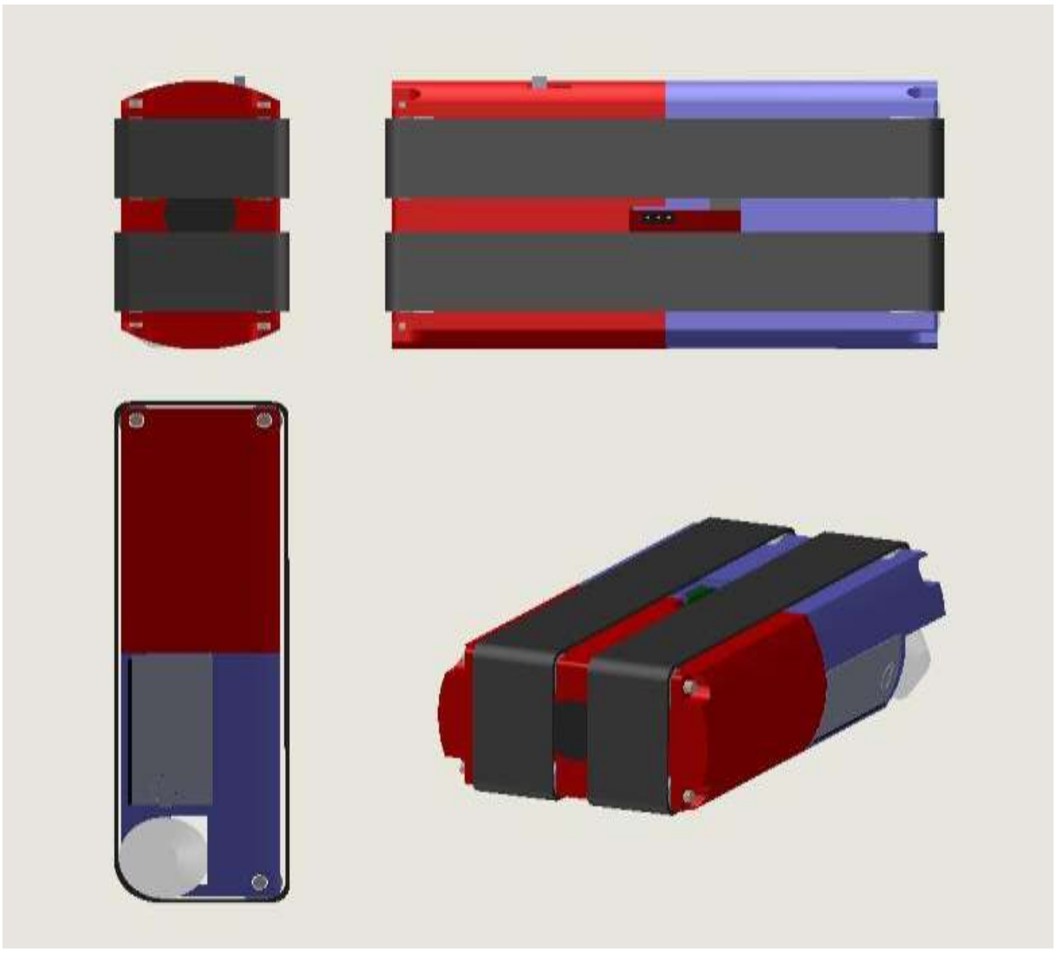
Appendix A: Sizes and Technical Drawings

A.1: Component Size List

Part name	**Dimensions (mm)**			**Notes on integration, sizing, orientation, position etc.**
	x	**y**	**z**	
Can size	66		115	66 diameter
Camera module & PCB	35	25	~3	Lens 16x16x22mm, on flex cable to Gumstix
GPS module	24	30	2x4	Two PCBs - GPS receiver itself and antenna. both ~ 4mm high
WiFi Module	30	8	4	Connected to the CPU PCB
CPU PCB	74.60	29.21	~5	
MCU PCB	46.3	25.4	~5	
Power PCB	40	34.9	6.5	Connected to the CPU and MCU PCB, as well as batteries
Gumstix	57	17	~3	Connected to CPU PCB
18500 Lithium batteries in holder	56	36	18	

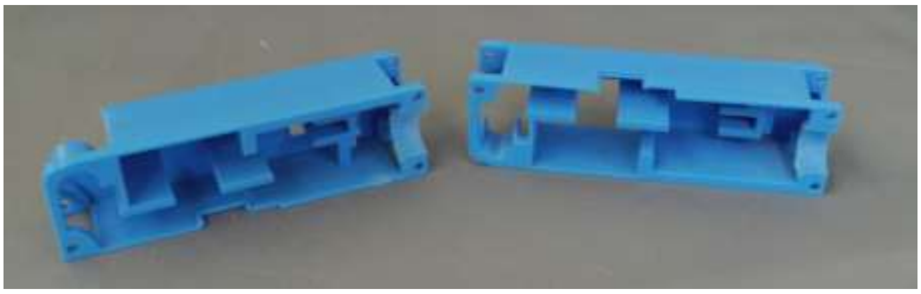
A.2: Technical Drawings

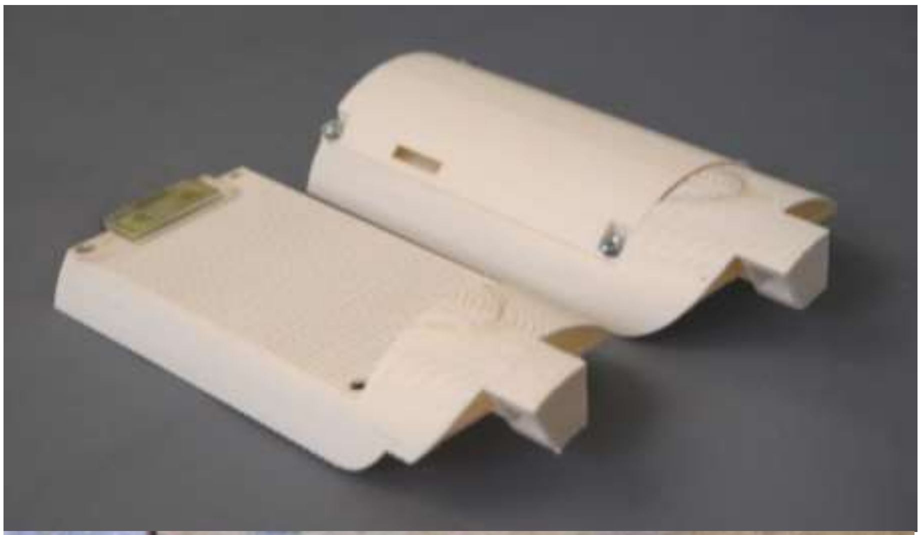


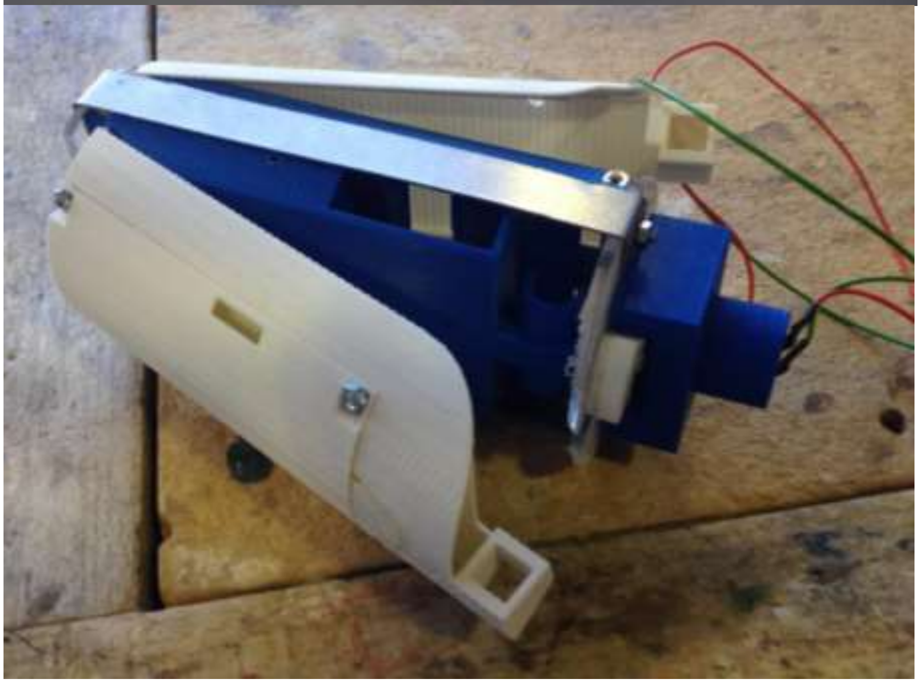


Technical note: The colours shown may not reflect the colour of the final device. They were used to aid in the modelling process.

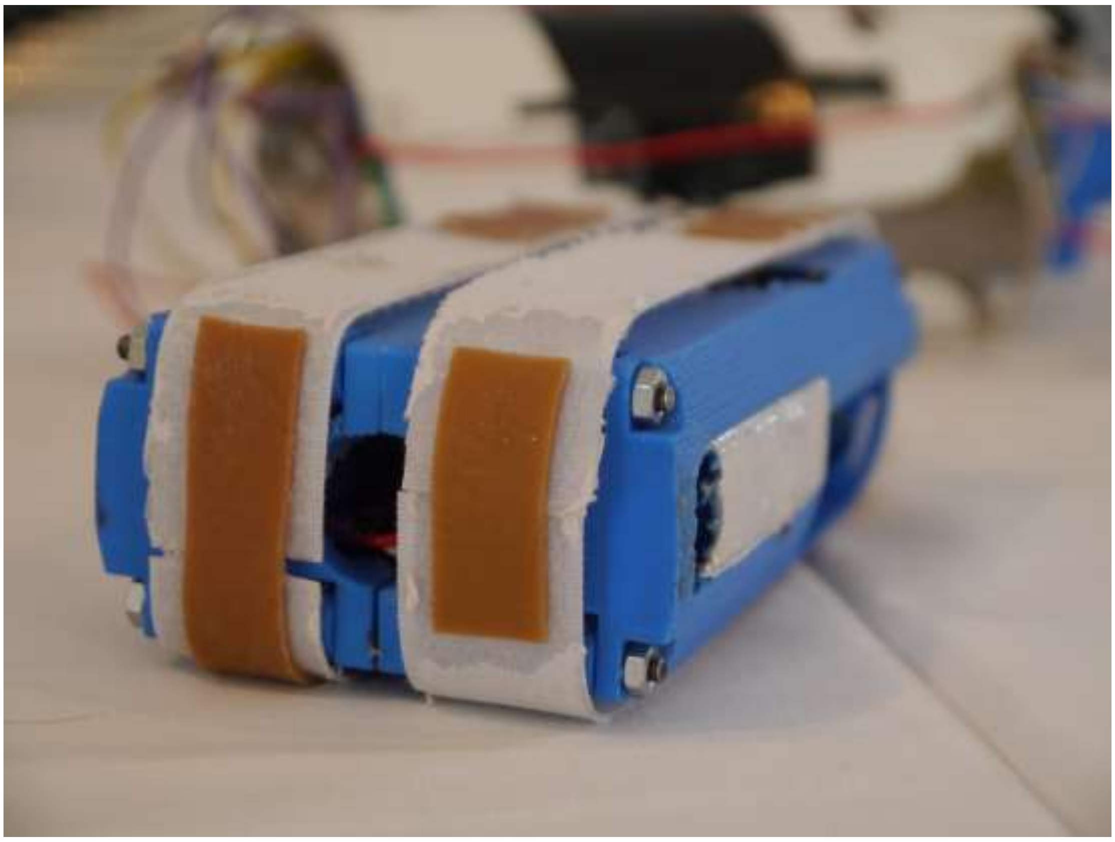
A.3: Photos of Rover Shell, Landing Module Shell and Dummy Assembly

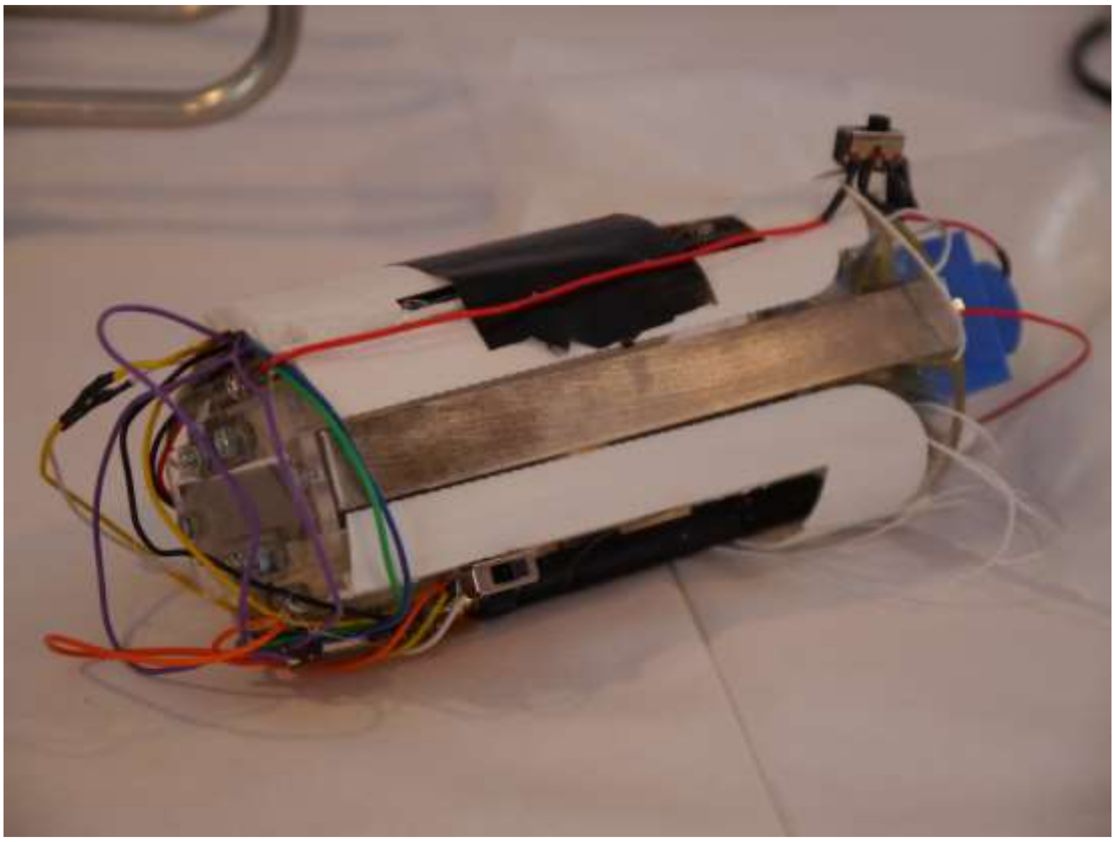






A.4: Photos of Final CanSat





Appendix B: Current Electronics Designs

***Technical note**: Two versions of each of the PCBs are shown. One is shown without the ground plane filled, to make the tracks easier to see. Another has the ground plane filled in, to show what the actual PCB will look. In some cases, tracks may appear to merge with the ground plane in pictures, but this is simply a side effect of exporting the PCB layout to an image. The original Eagle files can be downloaded from our Github here:*

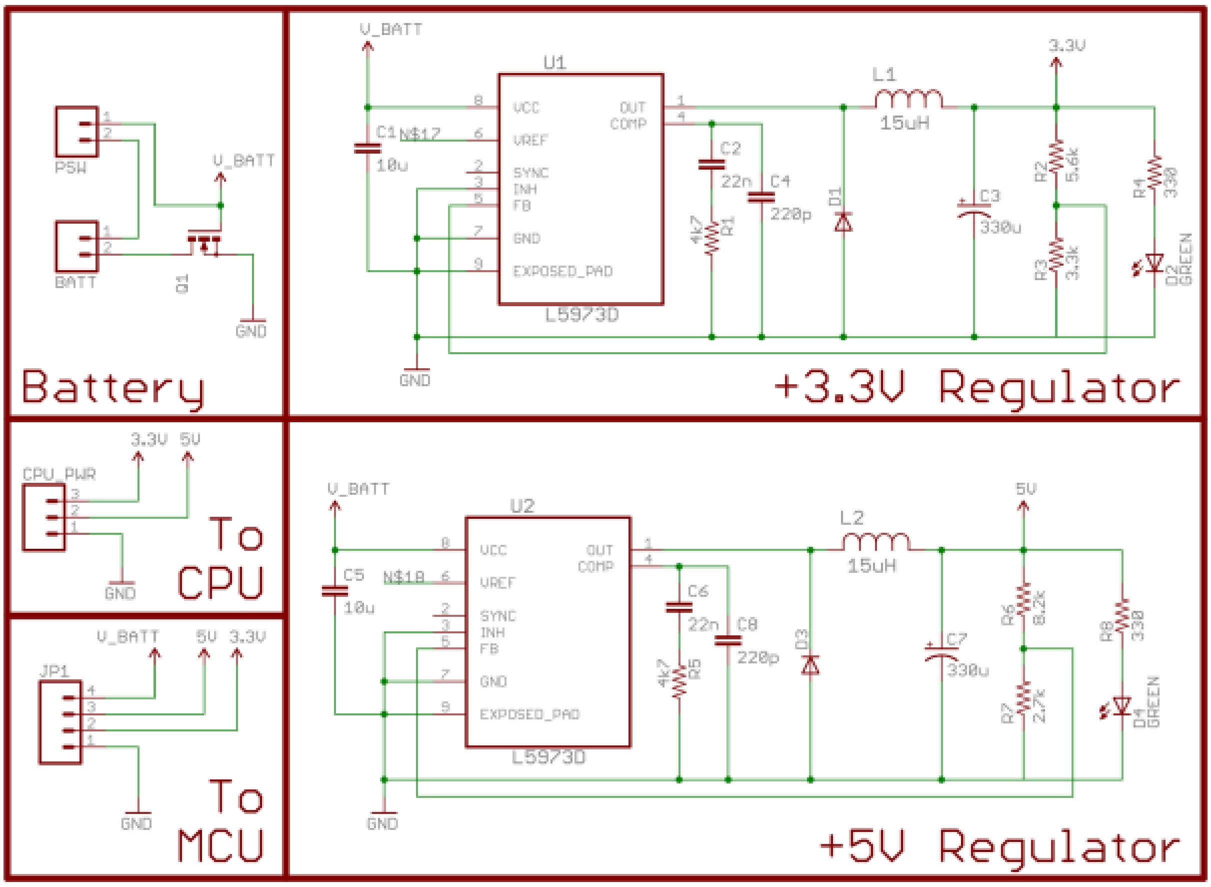
https://github.com/daveshah1/nova/tree/master/electronics

To view them, you will need to download the free version of EagleCAD, available at http://www.cadsoftusa.com/download-eagle/.

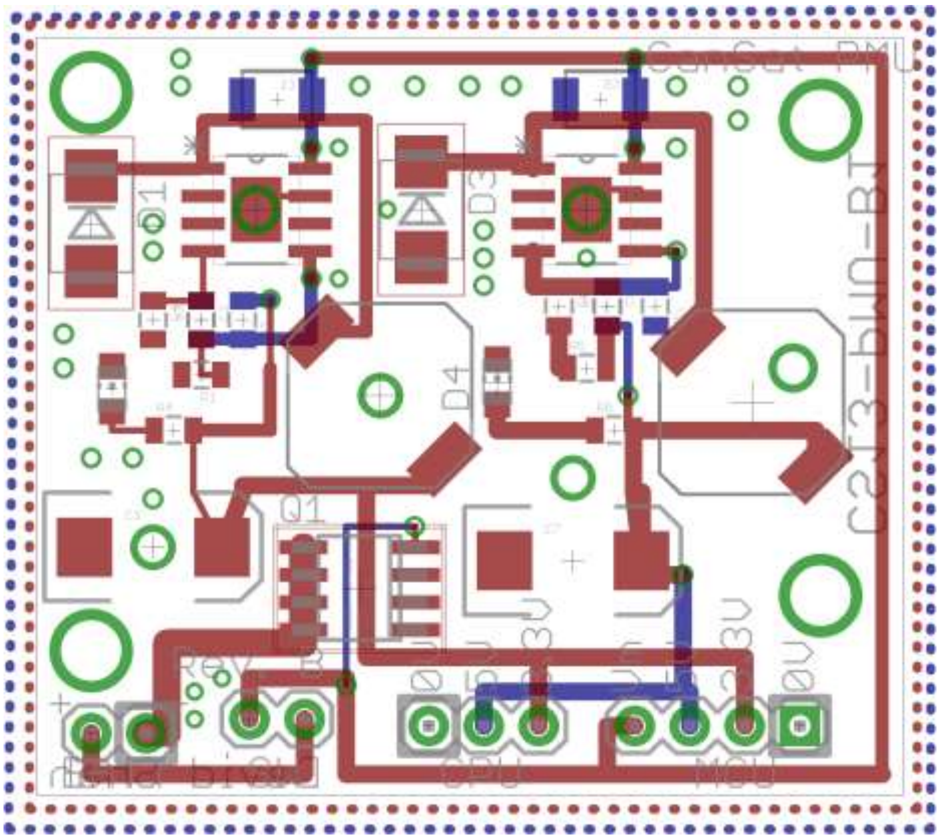
All the boards are sufficiently small enough that they can be edited with the free version. All content on the Git repository is freely available and licensed under the GNU GPL v2, as part of our contribution to the wider community.

B.1: PMU Board

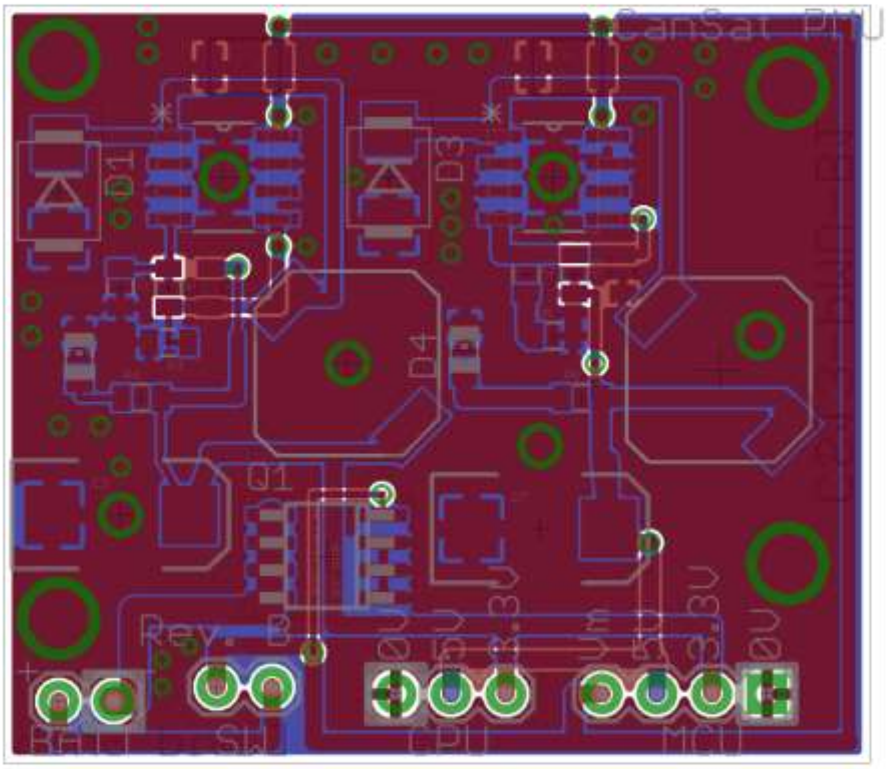
B.1.1 PMU Board Rev B Schematic Diagram



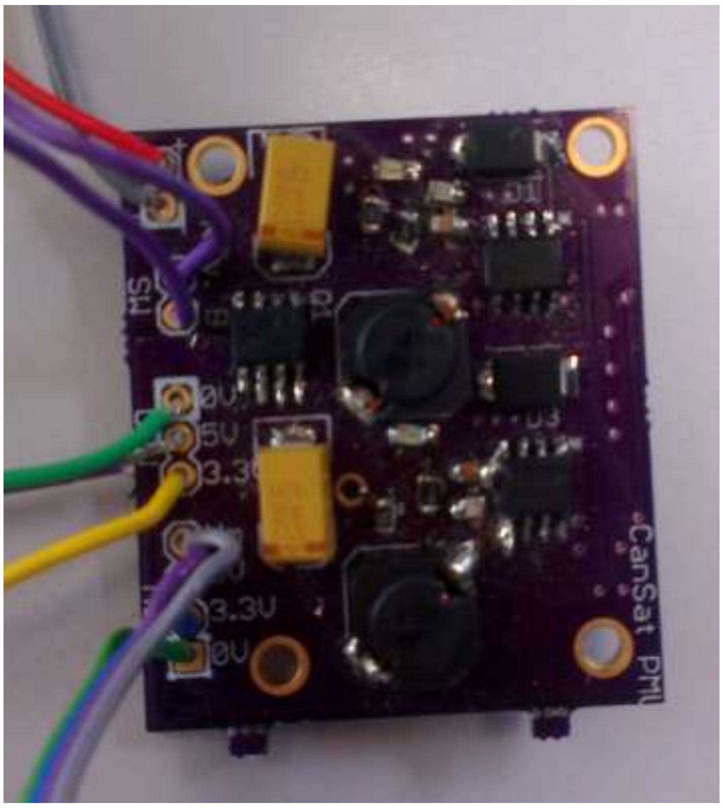
B.1.2 PMU Board Rev B PCB Layout (without ground plane)



B.1.3 PMU Board Rev B PCB Layout (with ground plane)

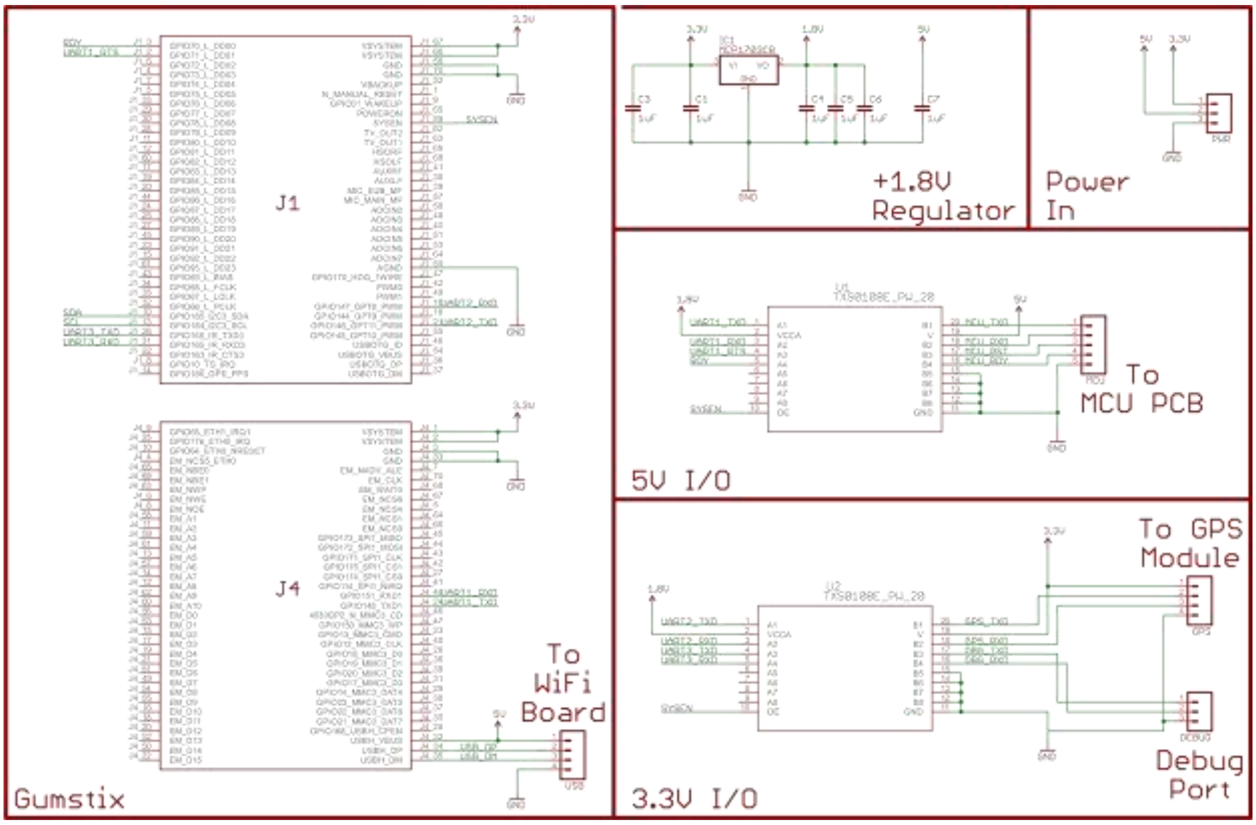


B.1.4 PMU Board Rev B PCB Photo

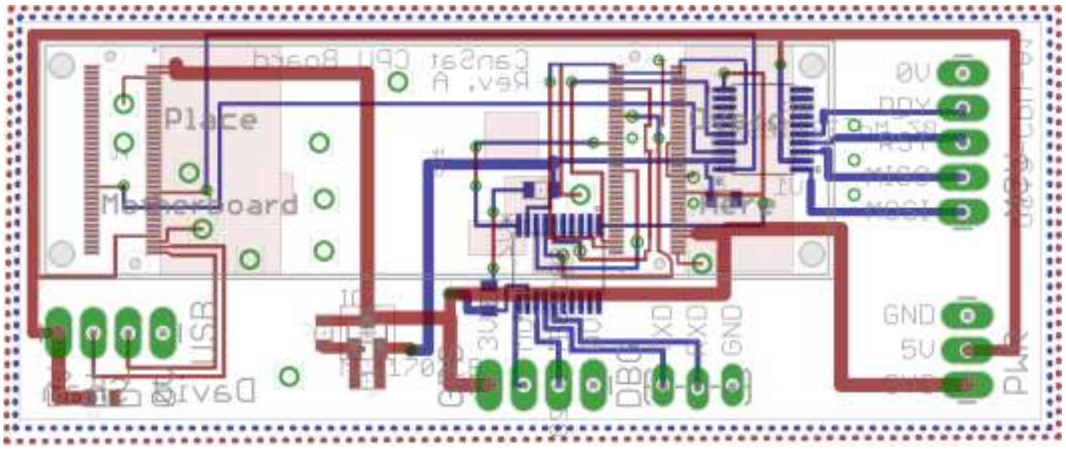


B.2: CPU Board

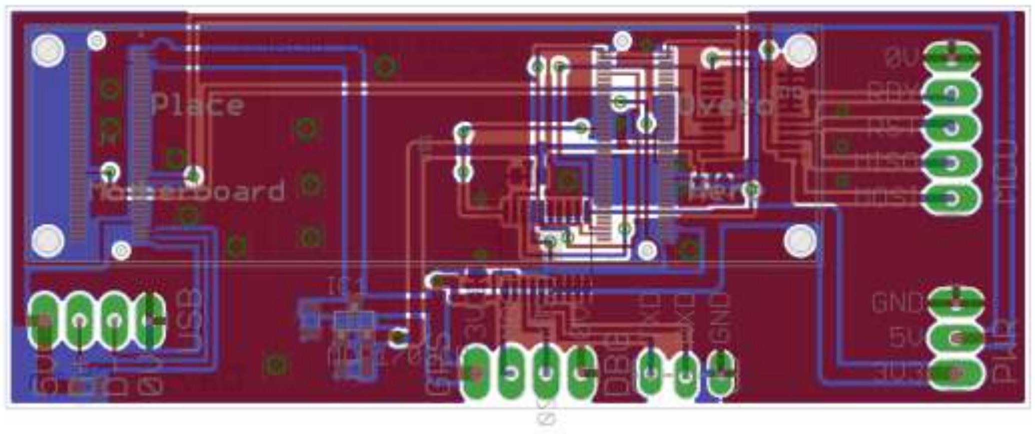
B.2.1: CPU Board Schematic



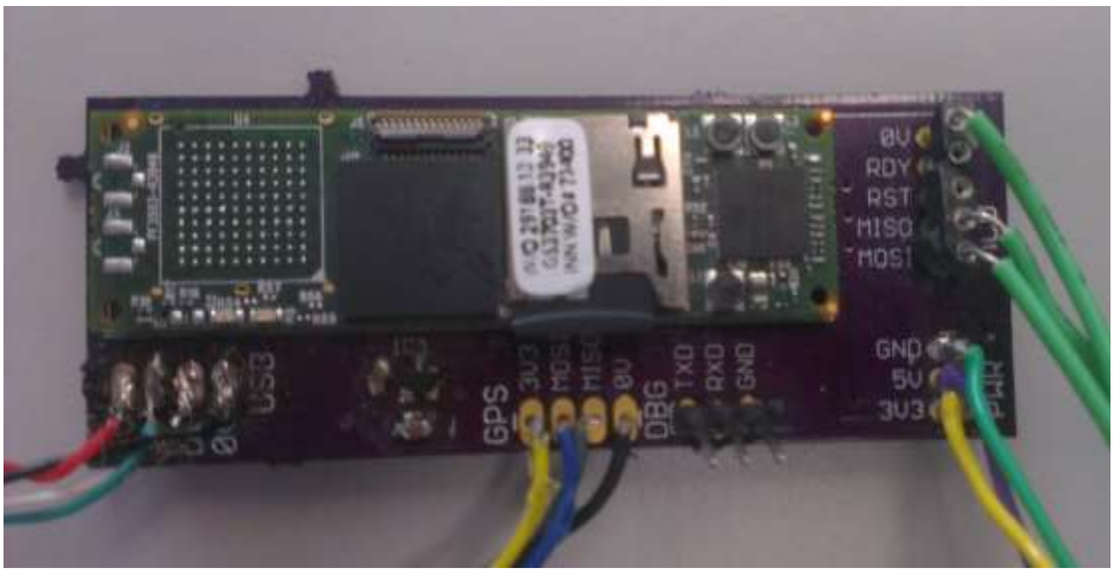
B.2.2: CPU Board PCB Layout (without ground plane)



B.2.3: CPU Board PCB Layout (with ground plane)

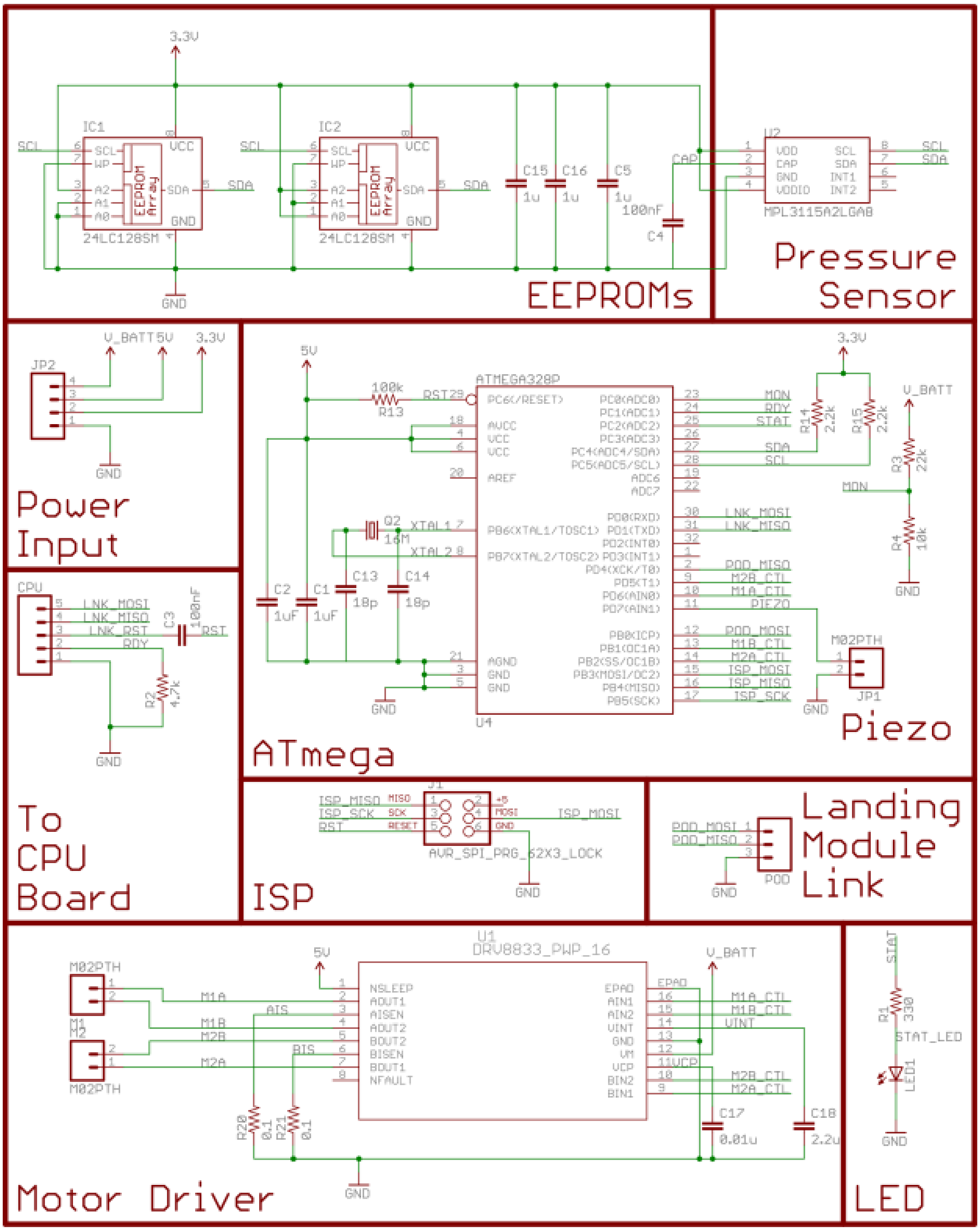


B.2.4: CPU Board PCB Photo

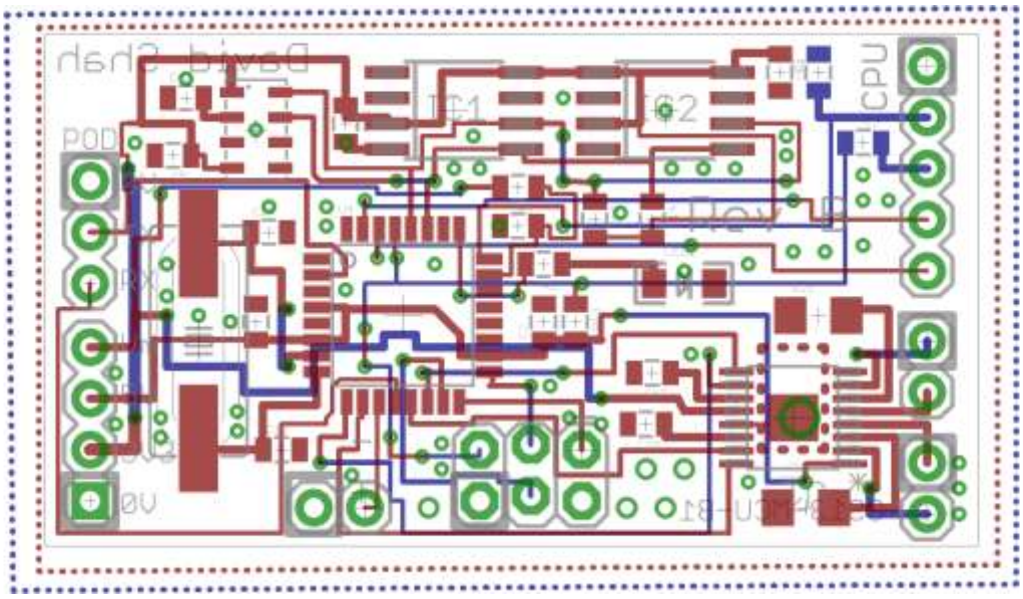


B.3: MCU Board

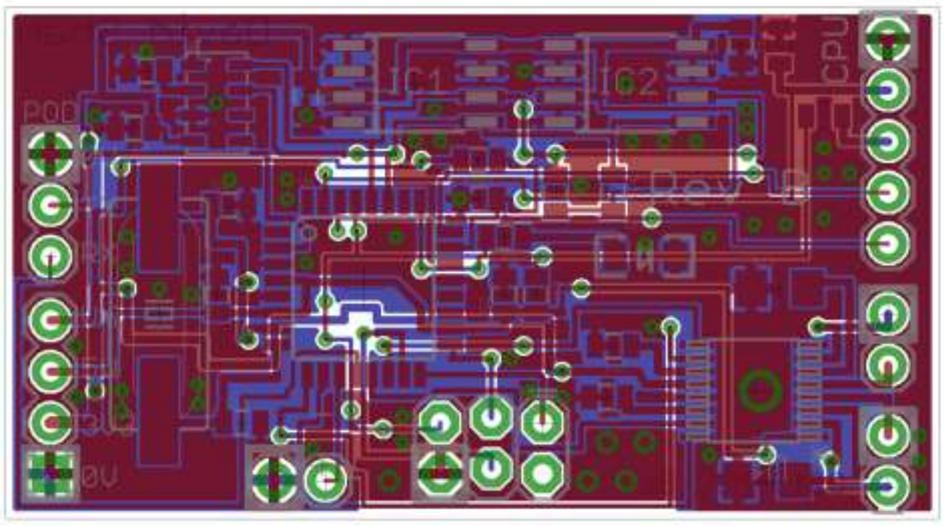
B.3.1 MCU Board Rev B Schematic Diagram



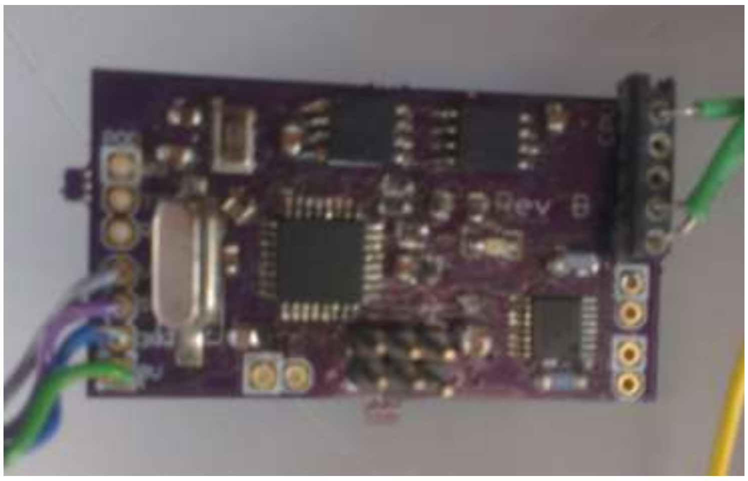
B.3.2 MCU Board Rev B PCB Layout (without ground plane)



B.3.3 MCU Board Rev B PCB Layout (with ground plane)

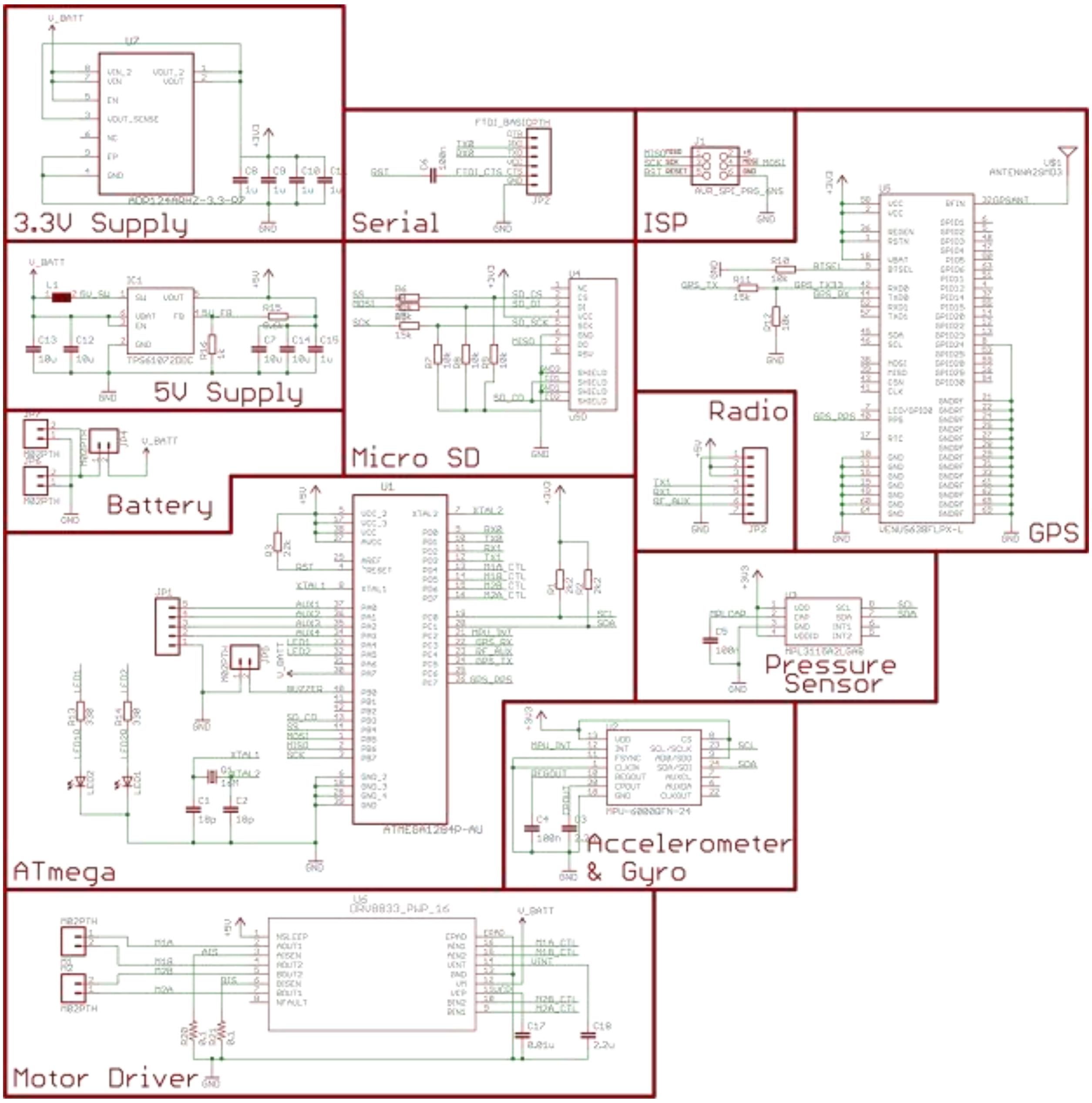


B.3.2 MCU Board Rev B PCB Photo

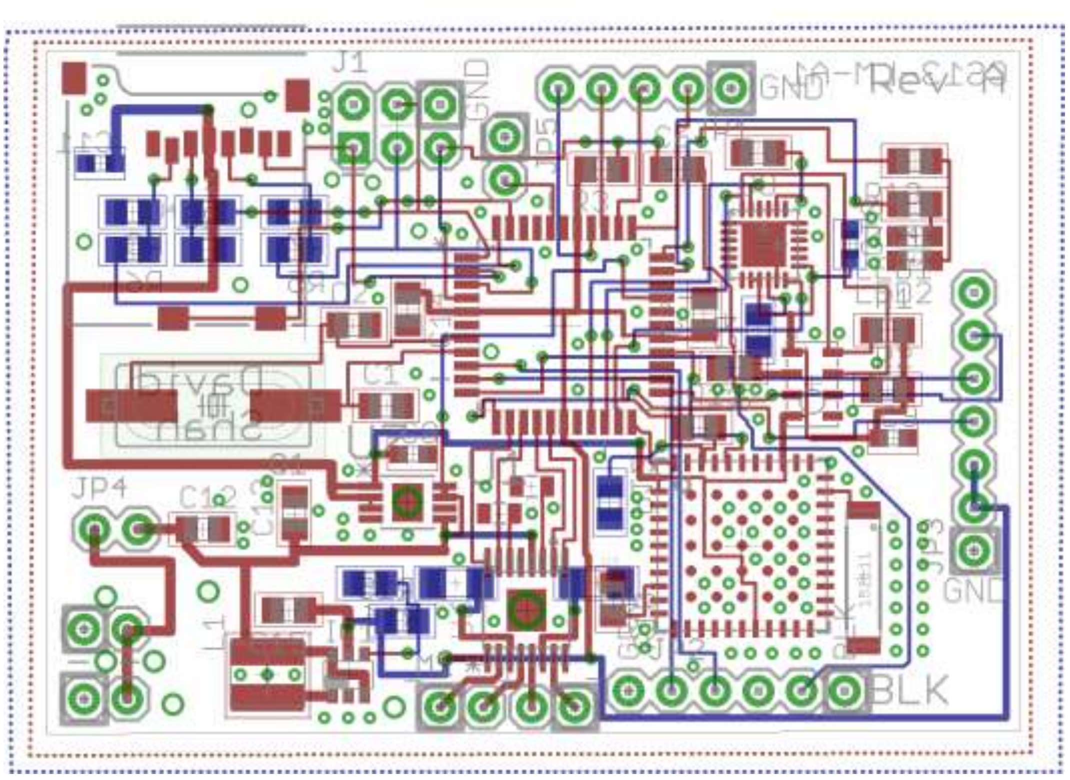


B.4 Landing Module

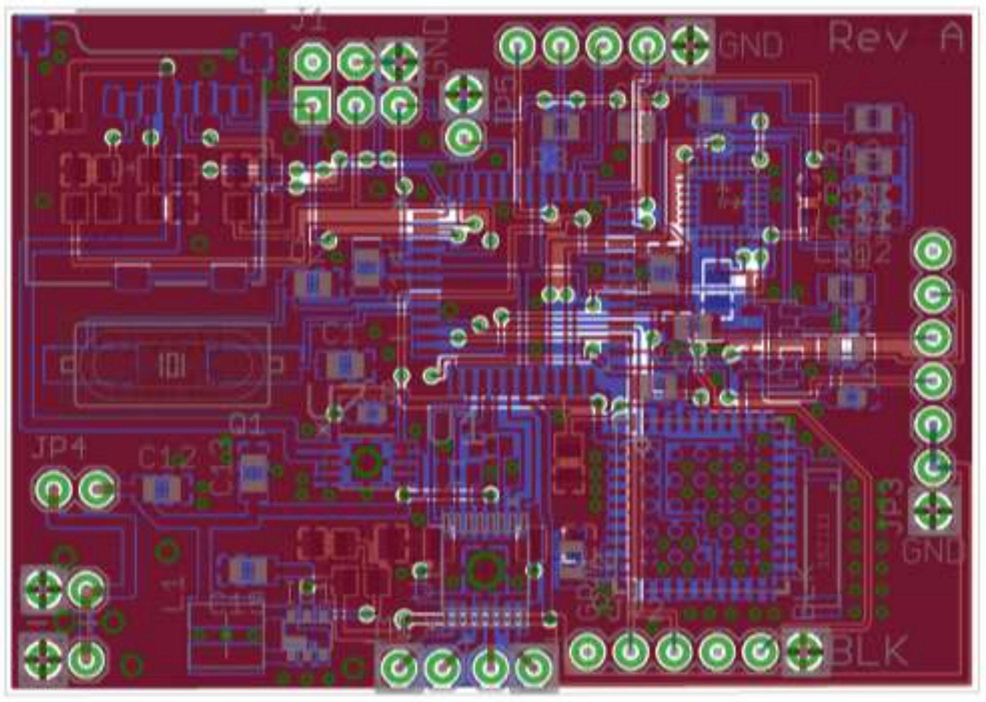
B.4.1 Landing Module Schematic Diagram



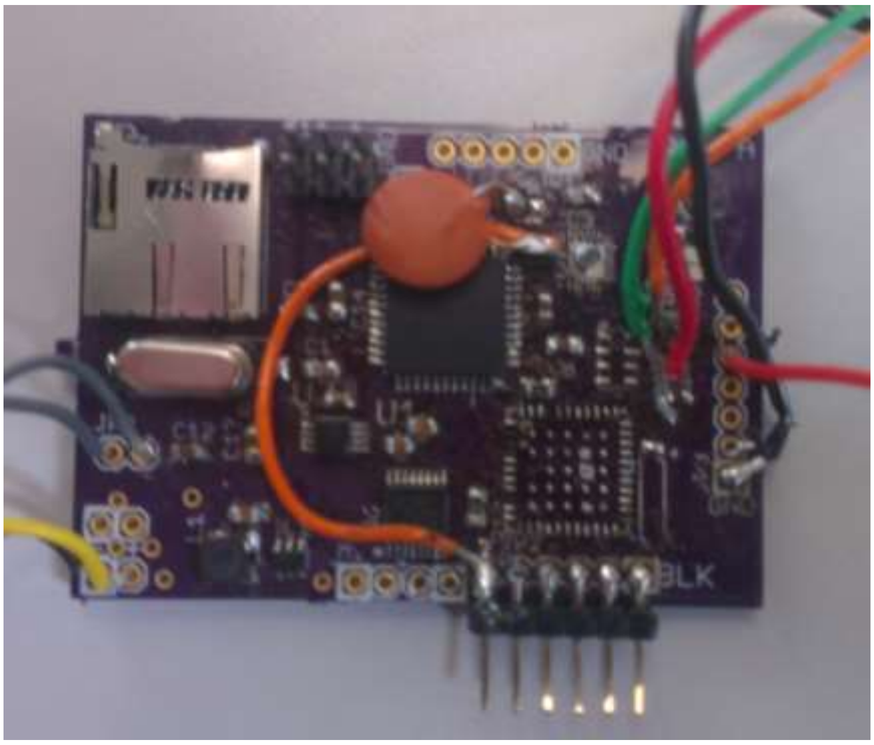
B.4.2 Landing Module PCB Layout (without ground plane)



B.4.3 Landing Module PCB Layout (with ground plane)



B.4.3 Landing Module PCB Layout (with ground plane)



Appendix C: Bill of Materials of CanSat

GRAND TOTAL: **£344.64**

C.1 Electronics Cost Breakdown

C.1.1 Rover

Product	**Cost (GBP)**
Gumstix Overo TidalStorm	88.04
Gumstix Caspa VL camera	47.16
Printed circuit boards (for 3 copies of each)	33.50
GPS module	13.75
MPL3115A2	2.01
ATmega328P (TQFP)	2.17
TXS0108E level translator x2	3.16
DRV8833 motor driver	1.88
L5973D regulator x 2	4.52
Miscellaneous components*	15.00
USB wireless dongle	7.50
24LC1025 EEPROM x2	4.82
Motors x2	6.00
Total cost	229.51

*Miscellaneous components include a range of inexpensive parts too numerous to list such as resistors, inductors, capacitors, diodes and connectors.

C.1.2 Landing Module

Product	**Cost (GBP)**
PCBs	10.98
DRV8833 motor driver	1.88
ATmega1284 microcontroller	4.96
GPS Module	13.75
Temperature/pressure and sensor board	10.79
Motor	3.00
Miscellaneous components	10.00
CanSat wireless module (estimated price)	20.00
Total Cost	75.36

Total cost of all electronics: **£304.87**

C.2 Hardware Cost Breakdown

3D printing costs calculated with use of HP DesignJet, at £0.20 per cm^3

C.2.1 Rover

Product	**Cost (GBP)**
3D Printed Chassis	15.82
3D Printed Idler Rollers x6	4.51
3D Printed Driver Wheels x2	1.92

Velcro strips	0.50
Total Cost	**£22.75**

*Miscellaneous components include a range of inexpensive parts too numerous to list such as rubber bands, and other mechanical components.

C.2.2 Landing Module

Product	**Cost (GBP)**
3D Printed Shells x2	10.02
3D Printed Motor Holder	5.70
Laser-cut acrylic	0.30
Nuts and Bolts	~1.00
Total Cost	**£17.02**

Total cost of all mechanical components: **£39.77**