

UK CanSat Competition

Team Orion

Alleyn's School

Final Design Review

Date: 17/04/2023

1 Introduction

1.1 Team Organisation and Roles

Member List and individual roles:

- C.L.
 - Designs and builds can, participates in outreach activities
- E.C.
 - Designs can, builds Yagi, builds parachute, participates in outreach activities
- I.J.J.
 - Designs and builds all hardware, builds can, participates in outreach activities
- M.K.
 - Organises meetings and has an overview of the project as a whole, ensures proper time-management and conducts risk analysis assessments, helps organise outreach activities
- R.V.
 - Presides over pecuniary decisions, organises all outreach activities, organises all marketing and promotion of project
- T.I.
 - Designs and creates GUI and app, participates in outreach activities

ROLE	NAME
PROJECT MANAGER	M.K.
PRIMARY MISSION HARDWARE	I.J.J.
SECONDARY MISSION HARDWARE	I.J.J.
CANSAT SOFTWARE	I.J.J.
DATA ANALYSIS + GUI	T.I.
APP DESIGN	M.K.
APP SOFTWARE	T.I.
CAN DESIGN	C.L., E.C. + All
CAN 3D MODELLING	I.J.J.
CAN MANUFACTURING	C.L. + I.J.J.
YAGI DESIGN + PHYSICS	E.C.
YAGI MANUFACTURING	E.C.
OUTREACH	R.V. + All
ACCOUNTANT	R.V.
MARKETING	R.V.

1.2 Mission Overview

1.2.1 Mission Objectives

Primary Mission:

Our Can will adhere to all of the CanSat requirements as detailed in the Competition Guidelines. To ensure this all team members have access to the requirements and bore them in mind when designing various aspects of the Can.

After release and during descent, our CanSat will measure air pressure, altitude, temperature and humidity (P.A.T.H.) and transmit the data once every second to the ground station where the data will be plotted and displayed as graphs.

Secondary Mission:

We decided that our secondary mission will be searching for habitability. We are simulating that our CanSat is on an exoplanet and we must try and figure out whether this planet would be able to sustain human life. To do this, in addition to the P.A.T.H. sensors we will also measure and transmit the concentrations of CO, NO₂, NH₃, NO, H₂, UV light and CO₂.

1.2.2 What will you measure, why and how?

Our CanSat will contain sensors which measure the pressure, altitude, temperature and humidity once every second in addition to sensors measuring CO, NO₂, NH₃, NO, H₂, UV light and CO₂. This data will then be sent by a radio module whose signal will be amplified by a Yagi antenna and the P.A.T.H. readings will be displayed in graphs and then used to determine the habitability of the exoplanet.

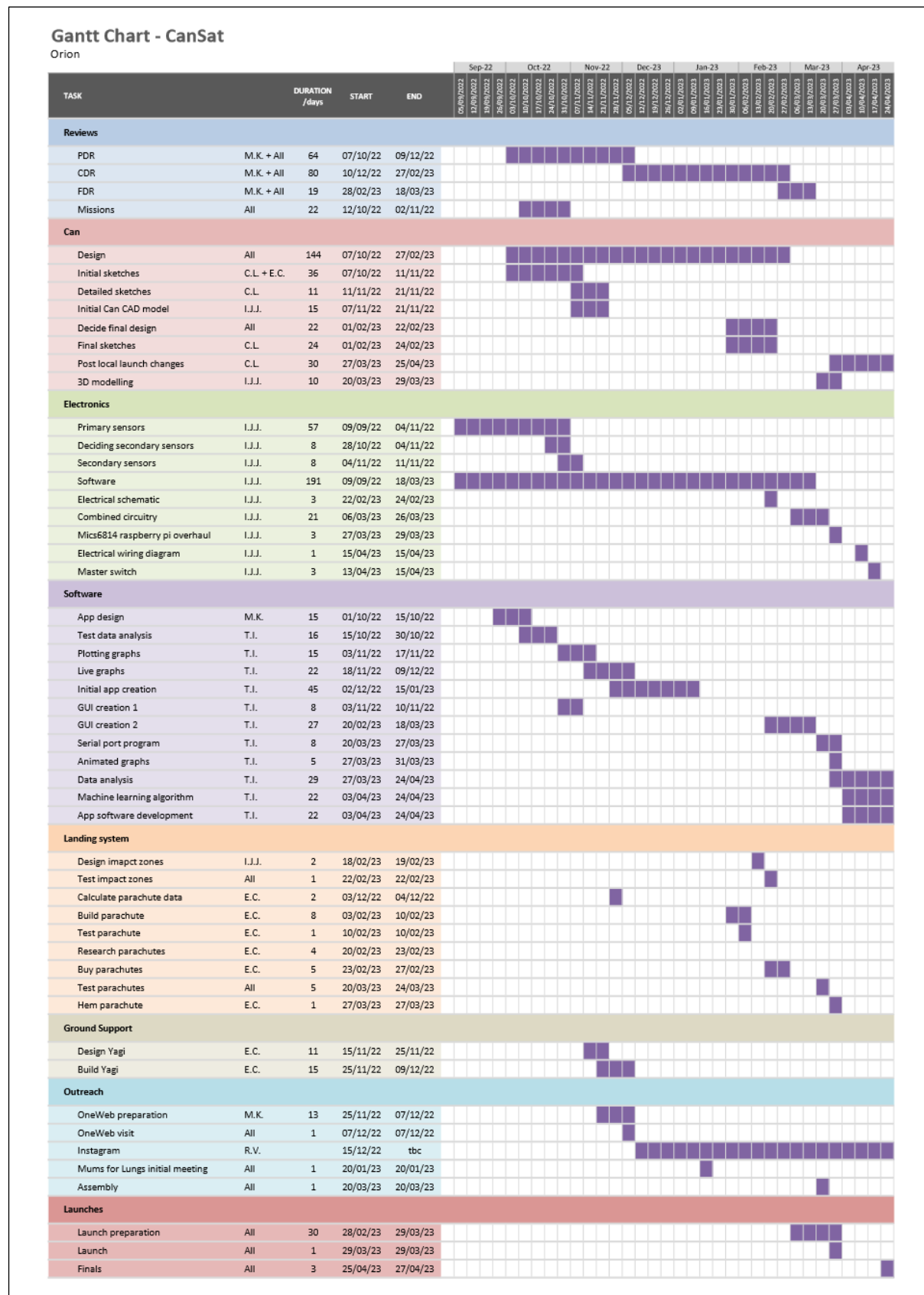
Habitability is determined by a program T.I. created (more detail is in Section 3.3.4) which compared the values received from the CanSat to known values to which humans can survive at. Given that we do not have an oxygen sensor and are testing the CanSat on Earth we are using altitude as a proxy for oxygen concentration. By the CDR the program was set to only compare the P.A.T.H. sensors but has now been extended for the secondary mission sensors as well.

We decided on search for habitability as our secondary mission partly because many of our team members set up and run our school's Astronomy Society and because we are all very interested in the possibility of humans eventually relocating and colonising other planets, the first step to which would be to find a planet where we could live. We were also inspired by the news of the ESA JUICE missions.

We decided to have our secondary mission as searching for habitability rather than life as we believe that that is more important and is more beneficial to humanity as a whole as any intelligent life that we come into contact with is unlikely to be friendly. We realise that given that our CanSat isn't actually going to an exoplanet the result will be that the Earth is, in fact, habitable, however still chose the secondary mission as we believe it is an interesting concept which will be used lots in the future when probes do go and explore exoplanets.

2 PROJECT PLANNING

2.1 Time schedule



2.2 Team and External Support

In order to identify what help we would need we first tried to identify what skills we each had. Even before doing this we quickly realised that we did not have anyone to help with marketing and outreach so asked R.V. to join the team. They would later also prove invaluable as they took on extra work as an accountant for our project.

To then identify our skills we made an Excel spreadsheet containing things like what A-levels we do, whether we did DT and/or Computer Science GCSE and other experiences that we might have.

Person	A-Levels						DT GCSE?	Computer Science GCSE?	3D printing experience?	Electronics experience?	Software experience?
	Maths	Further Maths	Physics	Chemistry	Economics	Art					
C.L.											
E.C.											
I.J.J.											
M.K.											
R.V.											
T.I.											

We concluded that overall we did have people with experience in 3D printing, electronics, software programming and design, however were lacking in general experience in team projects like CanSat as a whole. To remedy this we spoke to our school's team from last year to get advice from them. We also visited OneWeb (satellite company) to hear about real-world satellites and also listen to advice from the people there who had competed in a CanSat competition when they were at university. They gave us invaluable advice both specifically on our Can design and on our project as a whole. This included creating a mass and power budget, to test out different thicknesses when 3D printing the can and also change the infill percentage. They also warned that the sensors would almost certainly not give the gas concentration in ppm, rather just arbitrary values, so we would have to standardise those values against known concentrations of gas. For more details on our OneWeb visit see Section 4.

As our project progressed and we began to need to start buying various items such as sensors we realised that we needed funding. To do this we presented our project idea to our Head of School who was eager to provide us with the appropriate funding. Furthermore our Team Leader has proved incredibly useful to us as not only have they participated in the CanSat competition before so can provide invaluable advice, but also teaches the majority of the team Physics. This means that we are in constant contact with our Team Leader so can easily ask for help when needed and quickly get a response which has been very beneficial to us. We also received help from members of our DT department in building the Yagi, soldering the electronics and general advice on the CanSat as a whole.

2.3 Risk Analysis

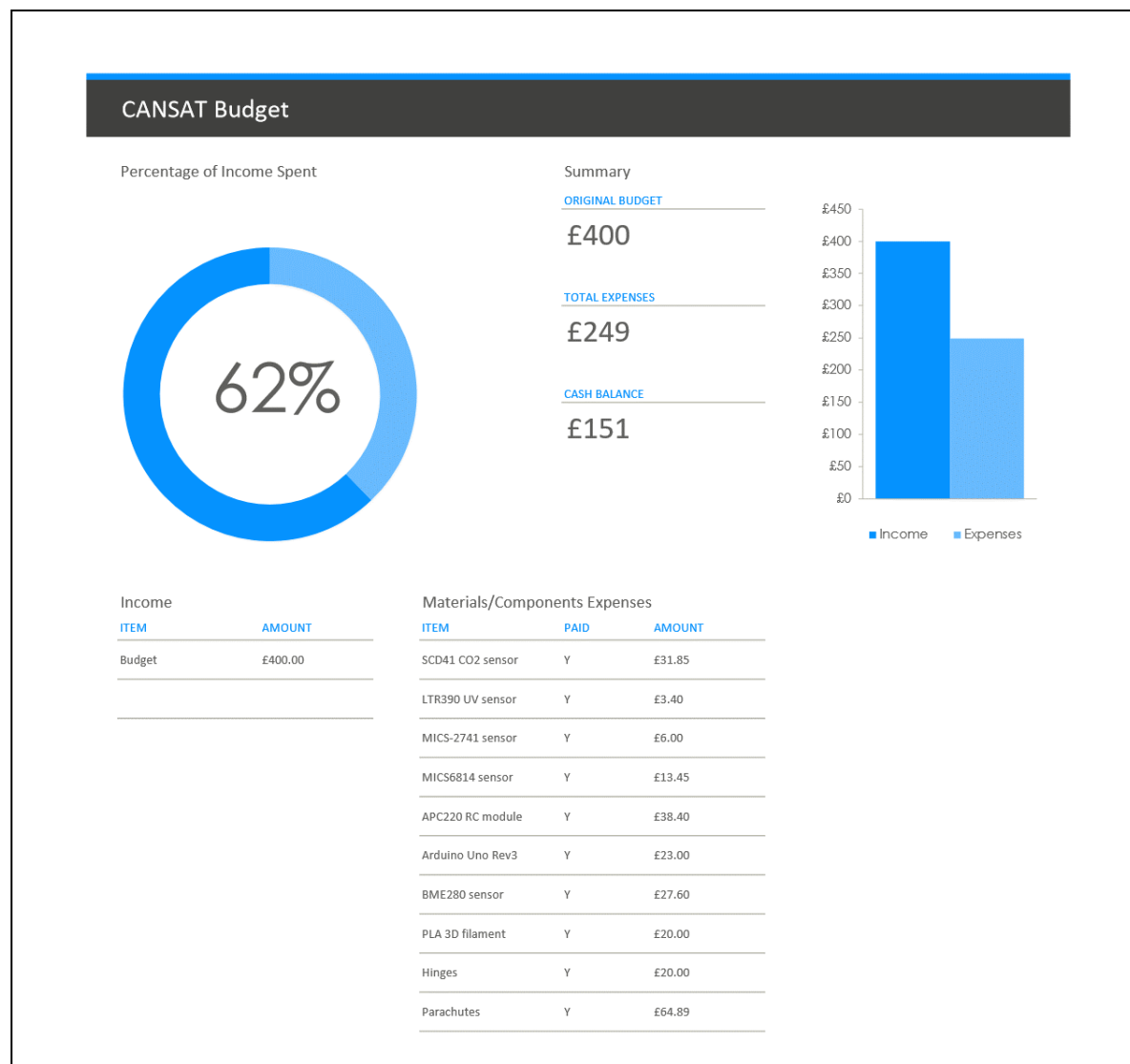
Our main concern was time management as a lot of our team are very busy. This meant that some weeks we were barely able to all meet up in order to further our progress on the CanSat. To try and resolve this we set up a Teams group chat for more effective communication between us and have now been able to meet much more regularly and consistently.

The main risk in terms of our Can used to be the legs not opening when the parachute deployed, however we have changed our design (more detail in 3.1.2) and are now much more confident that this will not happen. We are also concerned that perhaps the software

might not work on a certain laptop on the day of the launch so have pre-downloaded the tools required on 3 team members' different laptops just in case of failure / someone being ill on the day.

Luckily we are in constant contact with our Team Leader so any questions that we have are quickly answered, and due to our the structure of our school (having a joint 6th Form Centre) we are often in contact with those in the year above us both during breaktimes and in various clubs such as CCF and Model UN so we can easily access some more help if and when we need it.

Another aspect of our project that could go wrong was properly budgeting, so R.V. stepped up and acted as our accountant as well. The Can required many different parts which had to be bought online etc., which meant that we had to be realistic with our time and budget to make sure that we are organized and able to afford the parts we want. We created a budget spread sheet (which also displayed a colour coded pie chart for an easy, visual representation of our spending) so we would know how much of our budget was used up, why it was used, and how much we have left. As young people it is an important skill to budget our expenses together and to be able to be financially sustainable so it was incredibly beneficial to have done this.

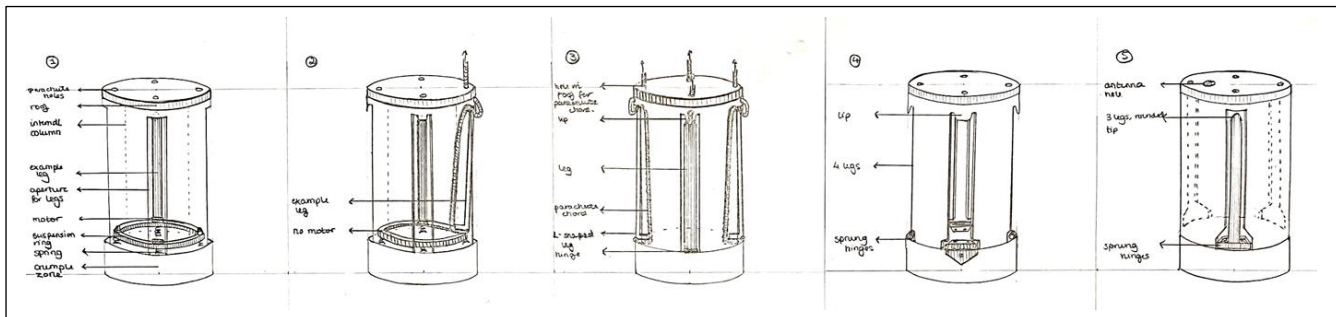


3 CANSAT DESIGN

3.1 Mechanical design

Throughout the CanSat project we have continually changed our design making new iterations as we test out different aspects of the Can. In the PDR we chose to have an internal suspension ring and have 4 support rods on the inside, however decided against this to ensure the legs and electronics would both fit in the Can.

See below for a timeline of our different designs:



Rather than being one continuous wall, the outer shell has a series of hexagonal holes throughout the structure. These holes serve two purposes: to reduce the overall mass of the satellite and to ensure the sensors within have greater access to the surrounding atmosphere. This ensures that the data collected during the satellite's flight is accurate, improving the useability of the results. The hexagonal shape of the holes was chosen due to the inherent strength of the hexagon. The fact that it is effectively composed of 6 equilateral triangles makes it the strongest shape in nature. Therefore, although the holes will compromise the integrity of the can's walls, the can will still be strong enough to withstand the force of launch and landing. When launching our Can there were fractures around the hexagonal holes, however due to the extra duct-tape the structure remained intact. We will therefore continue to use the hexagonal design and use duct-tape as extra security again.

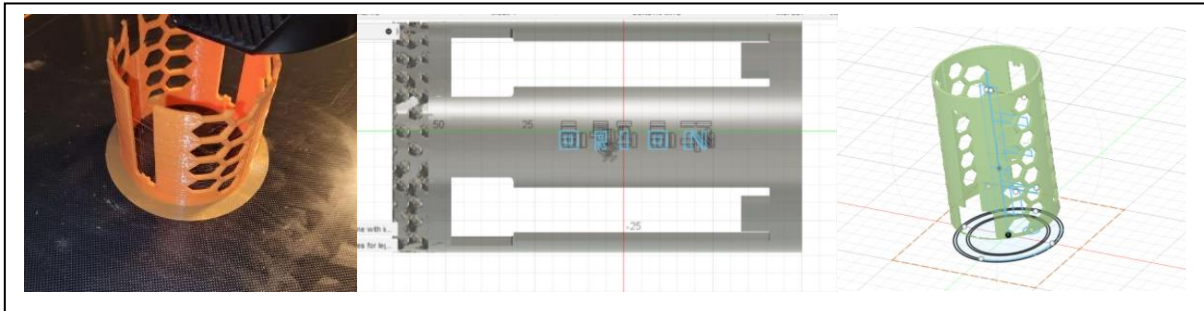
We settled on using sprung hinges to attach the legs to the base of the can. This means that the landing position is the natural position for the legs, so they are under tension when upright. Therefore, when released, they remain down throughout descent and landing. Before deciding to not use them for the regional launch we planned to keep them in an upright position by an 8mm overhang, which had the parachute cords threaded beneath it. As a result, when the parachute opens, the upward force on the cord would snap the overhang, releasing the legs into the landing position. We plan to use the legs for the finals.

After the regional launch we made various changes to the Can design.

PREVIOUS DESIGN	CHANGE MADE SINCE CDR	REASON
4 legs	3 legs	Offers more space for electronics when legs are in the upright position
3 grooves in base of can to store PCBs upright	Electronics loose in can but wrapped in cling film	Prevent water damage to electronics, ensures electronics do not fall out to leg apertures during launch and descent, prevents electronics from snagging on rocket during launch.
Hinges screwed into base of can	Hinges screwed into grooves in base of can	Reduces the amount of space the hinges take up in the can, increasing space for electronics

Antenna protrudes from hole in side of can	Antenna protrudes from hole in roof of can	Not enough space for antenna on side of can in the rocket at the local launch
Roof of can attached using superglue	Roof of can reinforced with duct tape	Upon consultation with the launch team at local launch we realised that the force of opening the parachute would snap the connections between the 3D Printed walls and roof of the can, and therefore the connections needed to be reinforced by duct tape

I.J.J. then modelled our Can(s) with Fusion 360 and 3D printed it. We also added an Orion logo into the side of the Can to make it more personalised.



3.2 Electrical design

3.2.1 Electrical design from PDR

When approaching the setup of the BME280 sensor, we ran into many issues - but resulting from the mistakes made, we managed to not only allow the sensor to function, but gained a better understanding of the way in which the electronics are wired. We first wired the Pimoroni BME280 breakout sensor to the Arduino using a breadboard for prototyping the circuit and then attempted to run a bit of test code. When run, this threw no errors, and was printing to the Serial port monitor, but only returned 0.00 double values. After a while of troubleshooting, we narrowed down the issue to two possibilities: either the port assignment was wrong, or the actual Arduino Uno was faulty. To solve the issue, we ran an I2C port checker (linked in Appendix 2.1), but this gave the same pin assignment as we had already been using (0x76), so we realised it couldn't be the issue. Ultimately, we switched the Arduino Uno board which we had been using to another one and this time, when the code (in Appendix 2.1) was run we received a feed of values which we then refined to send off in a particular format.

3.2.2 Progress made since PDR

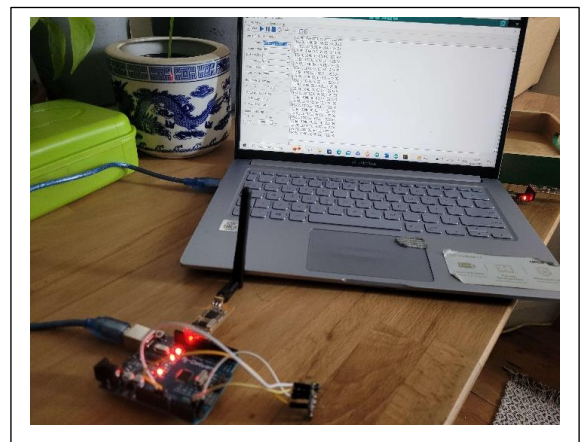
3.2.2.1 Finishing up primary mission electronics

After having successfully completed the electronics to satisfy our primary mission, the next step was to solder the BME280 sensor to its necessary pins on the Arduino Uno to avoid the issue of loose connections interfering with readings. The same code in our PDR was reused to test that the soldered connections worked and when run, yielded successful results.

3.2.2.2 Secondary mission sensors

DFROBOT MEMS Gas Sensor – MiCS-2714:

The purpose of the MiCS-2714 sensor was to measure the nitrogen dioxide and hydrogen levels in the environment of the Can. Nitrogen dioxide gas was chosen as a benchmark for habitability as nitrogen dioxide is a very potent and potentially dangerous pollutant here on



Earth, so in the event of exploring other planets, nitrogen dioxide would need to be checked as it is an acidic gas and can cause respiratory issues.

The first step we took was connecting the sensor to the Arduino using the wiring diagram provided by DFROBOT on their website¹, using a breadboard to test that the calibration code worked. We then continued by calibrating the sensor as detailed in Section 3.6.2.2.

PIMORONI MICS-6814 Gas sensor

As there were no online wiring diagram for the MICS-6814 sensor, we opted to design our own.² (See Appendix 2.2 Figure 1)

PIMORONI SCD41 CO2 sensor³

We then wired up the SCD41 sensor using Adafruit's wiring diagram as above, and repeated the same testing process using a breadboard as with the previous 2 sensors.

Running an I2C scanner found online, we found that the I2C address of the SCD41 CO2 sensor was 0x62 and made a note of this (See Appendix 2.2 Figure 2).

Having successfully found the I2C address of the SCD41 sensor, the next step was to calibrate the sensor, using the repeat measurement code, measuring temperature and humidity, provided by Adafruit, and have it calibrating for 5 minutes (See Appendix 2.2 Figure 3).

Adafruit LTR380 UV sensor

This sensor was connected using this wiring diagram⁴, showing that it connects via the I2C bus: SCL goes to A5 pin, SDA going to the A4 pin, GND to the GND (Ground) pin on the Arduino and the 5V pin going to the 5V pin on the Arduino.

3.2.2.3 Combined hardware circuitry

After testing the individual secondary mission sensors, the next task was to combine all I2C sensors in the final internal can electronics. Initially, the plan was to design an electrical schematic and have a PCB manufactured to reduce the likelihood of loose connections and having to resolder, whilst also reducing the space taken up by the electronics to free up room inside the can. However, in the interest of time and in an effort to save money in our budget, we made the decision to use small perfboards to create the combined circuitry and have all components neatly and compactly laid out inside the can. Having said this, since moving to perfboards, we created an electrical schematic (See Appendix 2.2 Figure 5) and, in the best case scenario, could resort to getting a PCB manufactured within the last few weeks.

The idea with the layout of the perfboard was to have the Arduino standing vertically within the can, and each perfboard vertically in parallel to the Arduino on either side, in order to maximise space. Notably, the initial design of having shelves to house each individual layer of electronics was revamped into this idea of vertical electronics as we concluded using the prototype printed in the PDR, that there would not be enough space in the can to accommodate the Arduino Uno microcontroller alone if placed horizontally, irrespective of the size of the PCBs on the shelves or the sensors themselves.

In order to do this, the electronics were laid out on the perfboard to gain a sense of where sensors would need to be connected and to arrange them in the best way possible to ensure that they took up as little space inside the can as possible. Once the layout had been finalised,

¹ https://wiki.dfrobot.com/Fermion_MEMS_Gas_Sensor_MiCS-2714_SKU_SEN0441

² Resources used: <https://github.com/eNBeWe/MiCS6814-I2C-Library/blob/43f8441ce9a3c3f2bb07ad50e4536cead32e9078/examples/Calibrate/Calibrate.ino> and <https://forum.arduino.cc/t/mics-6814-cjmcu-6814-gas-sensor-reading-conversion/992844>

³ <https://learn.adafruit.com/adafruit-scd-40-and-scd-41/arduino>

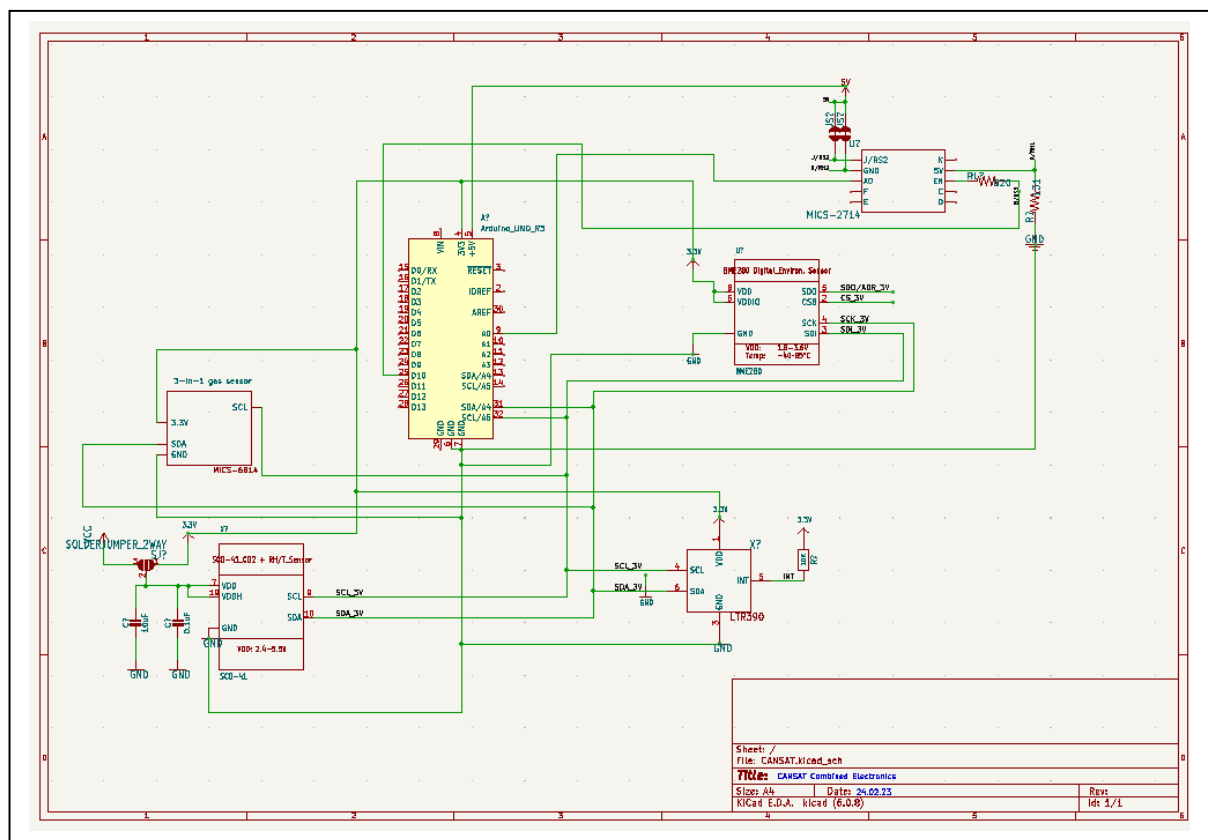
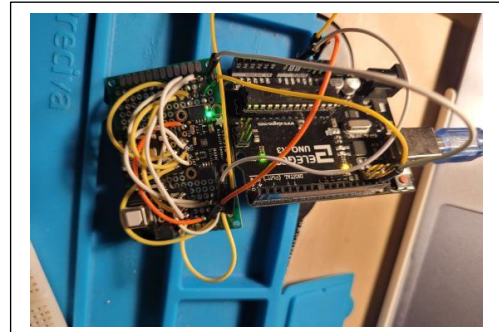
⁴ <https://learn.adafruit.com/adafruit-ltr390-uv-sensor/arduino>

the components were then soldered onto the perfboard, connecting individual perfboard contacts using solder to form a circuit which connected the MICS-6814 3-in-1 gas sensor, LTR390 UV radiation sensor and SCD41 sensor ('perfboard A') in parallel with the BME280 sensor of the primary mission using the I2C bus.

The MICS-2714 sensor and Adafruit GPS module were connected individually on a separate, smaller perfboard ('perfboard B'), with the MICS-2714 sensor occupying the A0 pin on the Arduino, and the GPS module using the RX (receiver), TX (transmitter) pins. (See Appendix 2.2 Figure 4)

Below is an electrical schematic of the sensors, omitting the APC220 radio module and the GPS module. It shows that the MICS6814, UV sensor and CO2 gas sensor are connected in series, with the BME280 sensor in parallel to them, and the MICS2714 sensor is connected separately.

See right for perfboard A being tested and see Appendix 2.2 Figure 4 for perfboard B being tested.



3.2.2.4 Changes since CDR

A conclusion was made that the MICS6814 3-in-1 gas sensor we had selected didn't have full online support for Arduino. As a result, we decided to attempt to overhaul part of the secondary mission electronics (the MICS6814 gas sensor) to a raspberry pi nano, which

would be connected to the Arduino. This wasn't possible within the time frame of the regional launch but was started in the final amendments section. This was important as the MICS6814 sensor is important for measuring 3 gases: nitrogen dioxide (oxidising), ammonia and carbon monoxide (reducing).

Another decision was made to no longer include the MICS2714 nitrogen dioxide sensor as the MICS6814 measures the same gas. And the LTR390 UV radiation sensor was left as a final addition to be made after the MICS6814 was overhauled to the raspberry pi, which is still an ongoing effort.

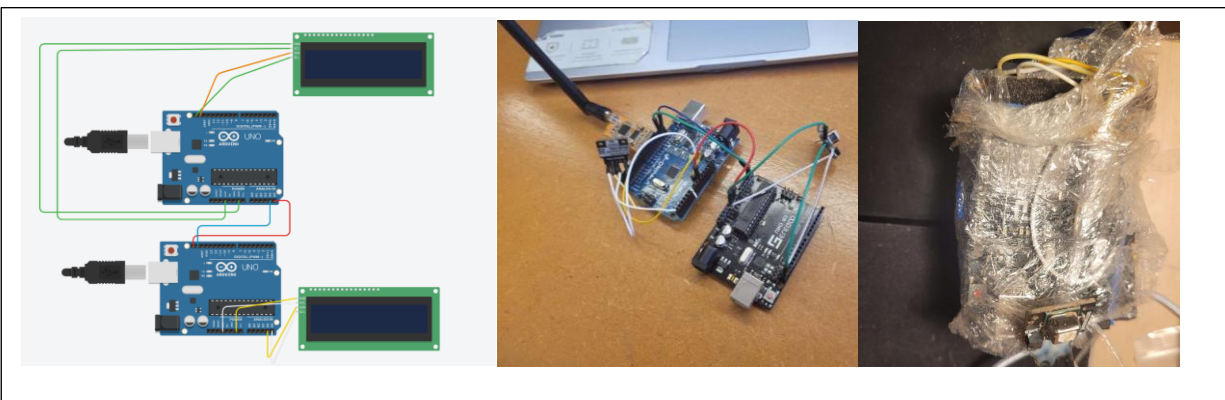
Furthermore, in the interest of time and getting things prepared for the launch, we opted for a 2-arduino electronics system where the master Arduino carried out the primary electronics mission, and the second Arduino carried out the secondary electronics mission. The master Arduino then sent data via the radio module to ground station. The actual connections between individual components on each Arduino remained unchanged and still implemented I2C interface for the regional launch. Finally, previous ambitions to manufacture a custom PCB did not come to fruition, in favour of focusing time and resources on perfecting the functionality of the electronics.

3.2.2.5 Complete electronics

The final step was to assemble the entire electronics together and have elements of the secondary mission work alongside the primary mission. For the purposes of the regional launch, we decided that in the interest of time and getting the readings we needed, that it was important to test the current carbon dioxide sensor for the secondary mission.

The electronics were wired up as shown below for the launch, with each connected to a separate 9V battery. In this way, the primary mission electronics and software were localised on the master Arduino, and the secondary mission electronics were localised to the second Arduino which continuously fed data to the master Arduino.

The electronics were also cling-filmed to prevent short circuits and better protect the electronics.



(Above left is a tinkercad circuit design made to show the main Arduino (below) and secondary Arduino (above). The below and above OLED display components are placeholders and represent the BME280 and SCD41 I2C devices respectively. The APC220 RF transmitter module is omitted.)

The antenna for the APC220 RF module was soldered onto the back of the Arduino to minimise space taken up, so the aerial stuck out of the side. A short-term fix was to strip the antenna so that it took up less space. However, it sticking out the side still caused issues when

fitting the CanSat into the nose cone as it was a tighter fit than intended, so for finals this aerial will be soldered vertically instead.

Additionally, the use of 2 separate 9V batteries took up too much space in the can so we decided not to implement the legs yet for the purpose of the regional launch. In future we will likely use a flat battery pack and one that is more compact. The overhauling of the MICS6814 to a raspberry pi detailed later on in this write-up would also mean that only one battery, connected to the raspberry pi, would be needed for both the Arduino and raspberry pi.

Attached is a video with the full working electronics from the regional launch:



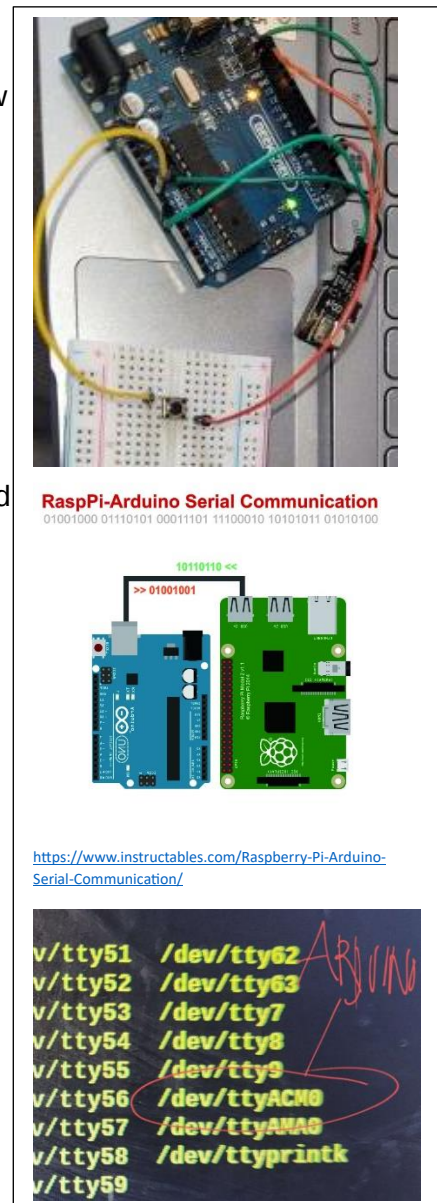
CANSAT REGIONAL LAUNCH.mp4

3.2.2.6 Final electronics additions

A master switch was implemented to ensure that the Arduino would reduce power consumption until its launch, as during our regional launch in Chelmsford, we realised how important it was to conserve the battery until launch. As Arduino has no built-in method of coding the microcontroller to shut down, we opted for implementing a master switch that would toggle low power mode when pressed. Using the Low power library in Arduino, whenever the pushbutton is pressed, it would send a value of HIGH to the microcontroller, which when checked using a conditional statement, causes the microcontroller to enter low power sleep mode and reduce power consumption so no output is made until the pushbutton is toggled again. This means that during finals, the electronics can be powered off until the point when they are launched. In order to get the master switch to work across microcontrollers using bidirectional serial communication, the next step is to update the code to write the Boolean value to the raspberry pi that reads false if the button switches the circuit off, and shut down accordingly when checked by the raspberry pi.

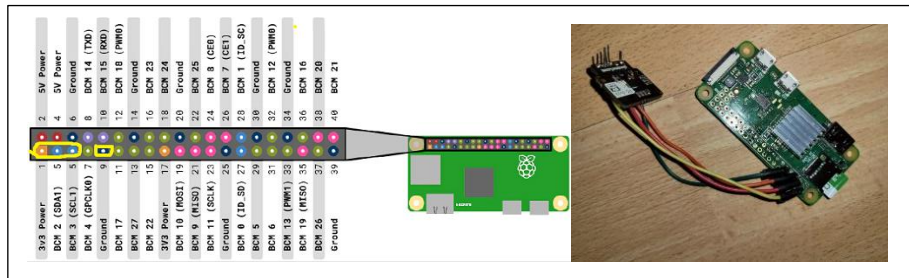
Having completed the regional launch, the issue of space within the CanSat needed to be addressed. The use of two Arduino Unos as a solution to connecting multiple I2C sensors was ideal for power budget, but an issue for space and efficiency. Resultingly, we are attempting to implement the secondary mission MICS6814 gas sensor on a smaller raspberry pi nano and replace the second Arduino with a parallel I2C connection between the BME280 and SCD41 sensors using two 10k pullup resistors.

To the right is the wiring diagram for the proposed setup.



The progress made so far has allowed us to connect an Arduino connected to a SCD41 sensor in serial to a raspberry pi nano, which itself is connected to the MICS6814 sensor. As seen in the command line terminal, the raspberry pi recognises the Arduino as a serial device:

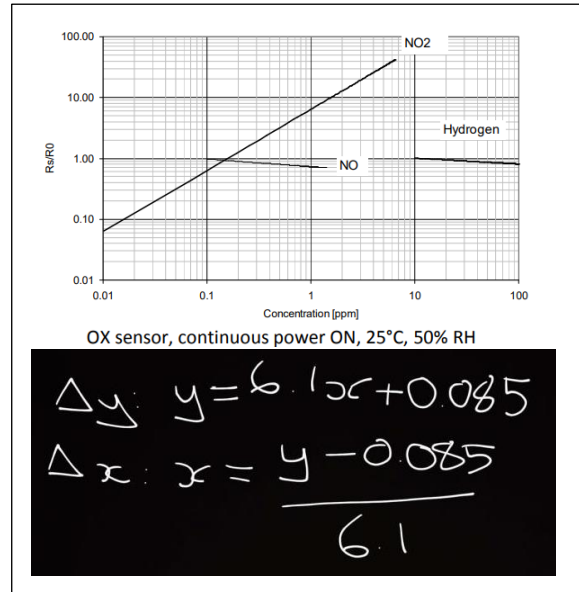
The soldered raspberry pi is as such, making use of the I2C bus 1 of the raspberry pi:



The example code from pimoroni's GitHub library was then uploaded to the raspberry pi using the Thonny IDE to test whether the I2C connection functioned or not.

[\[https://github.com/pimoroni/mics6814-python\]](https://github.com/pimoroni/mics6814-python)

Finally, in order to format the MICS6814 readings as concentrations in ppm rather than resistances based on voltages, using Pimoroni's datasheet, calculations, such as the one below for oxidising gases (nitrogen dioxide) were be carried out in the python code to convert these values to concentrations. For example, the gradient was calculated approximately from the graph to be: $(1-0.085)/0.15 = 6.1$.



3.2.2.7 Electronics post-launch conclusions

It is evident that we now need to fully implement the overhauled secondary mission electronics alongside the primary mission electronics by sending data across from the raspberry pi nano to the Arduino uno using USB connection ahead of our launch at finals.

Furthermore, code needs to be updated for the MICS6814 sensor to format it into the format used by the rest of the electronics in the on-board software.

It was clear after the launch of the CanSat that the antenna needs to be resoldered to fit vertically and stick just through the lid as opposed to sticking out of the side.

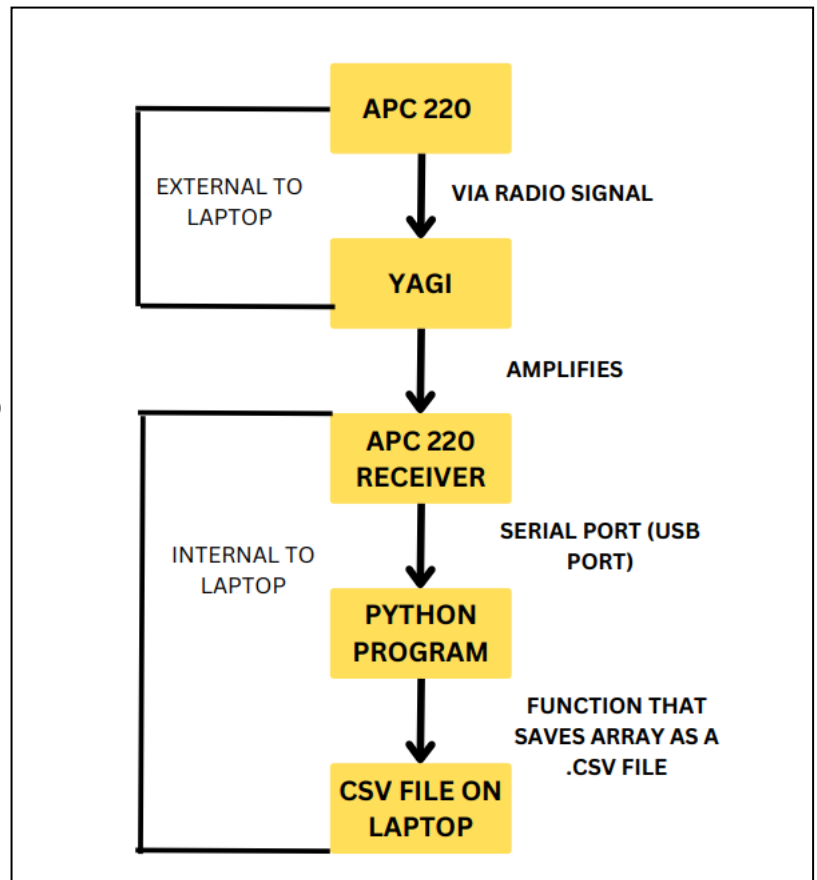
Finally, the master switch code needs to be updated for the final layout of sensors across the Arduino (connected to the BME280 and SCD41 sensors) and the raspberry pi nano (connected to the MICS6814 3-in-1 sensor), by periodically sending a Boolean variable to the raspberry pi from the Arduino using bidirectional serial data transfer. And the pushbutton must be soldered onto the Arduino.

3.3 Software design

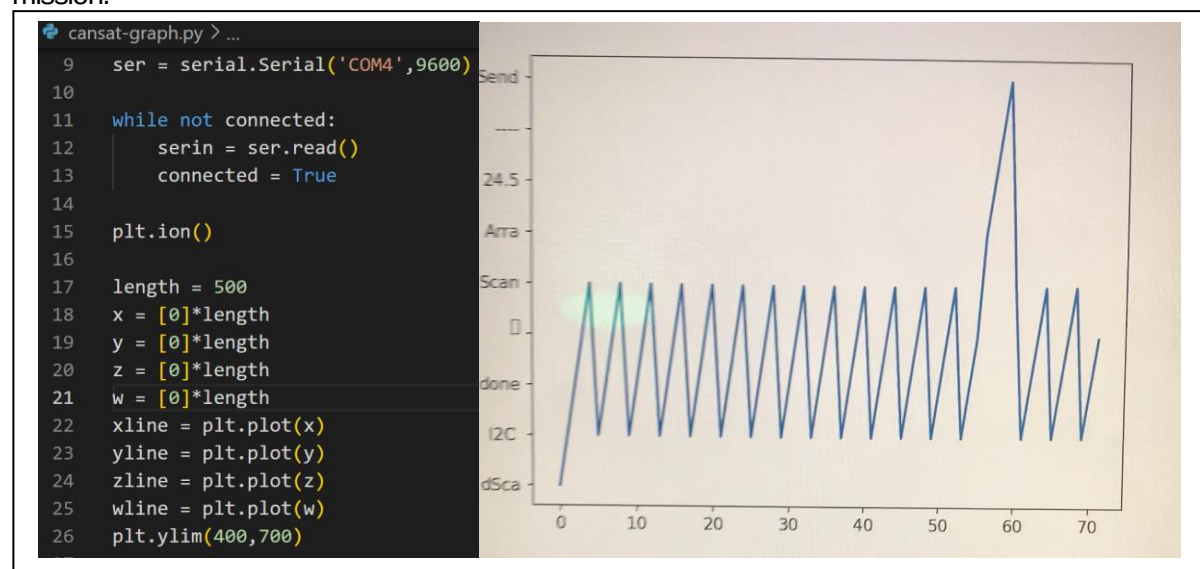
3.3.1 Live graph

Before we can analyse the data and complete our secondary mission, the data from the sensors attached to the CanSat must first be transferred to the laptop that the python program runs on. The Arduino which the sensors are attached to transmits a radio signal via the APC 220. This is then amplified by the Yagi which helps the signal to be sent to the APC 220 receiver which will be plugged into the USB port of the laptop. Then using the pyserial module to open up the serial port and create a program which stores the data into different arrays, a live graph is created which displays how the different P.A.T.H. values vary with time for the duration of the CanSat's journey.

Previously, the rate at which the program drew the graph was too slow making it difficult to draw conclusions about the CanSat's journey. This was improved by adjusting the time that each data point is plotted so that the graph appears to be plotted instantaneously.

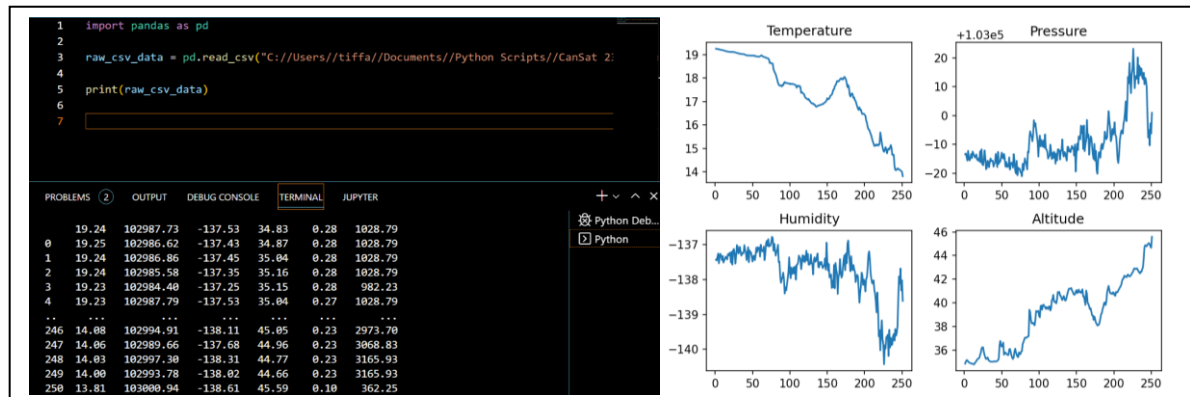


When creating the live graph many variables had to be created in order to access the data, since the Arduino sends and receives data in a specific manner. The python program uses pyserial to open up the COM4 port at baud rate 9600, which means the computer can actually receive the data. After this, it indexes the data from each item in the list (which gets updated each second) and decodes it so that the numbers are in base 10. Finally, a new array is made which then adds each of these data points and updates the graph per second. When the program is terminated the variable `is_running` is set to False, which causes all the arrays of data to be saved into a csv file for later use in the GUI and secondary mission.



3.3.2 Data processing

We decided to then convert the data detected by the Arduino's sensors into graphs and create a Graphical User Interface (GUI), in order to make the user's experience with the data more interactive and engaging. The data collected from the process explained in Section 3.3.1 then needs to be manipulated and collated. The image below shows what the .csv file looks like. To make the data more visually appealing we needed to create graphs. We created a function which converts data from a .csv file from an Arduino into 4 plots (graphs) for the P.A.T.H. variables each against time, which can then be used and displayed in the GUI.



The pandas package is a popular resource for data science projects, hence we decided to use it for ours as well. The package inherits many properties from NumPy, another python package, and this allows programmers to conduct operations on arrays.

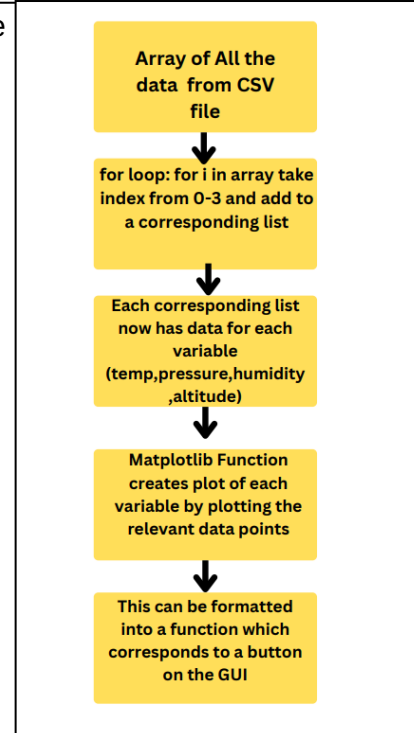
A variable "raw_csv_data" is created and converted from pandas.core.frame.DataFrame to an array, using the to_numpy() function. After this, an empty array for each of the variables is created and iterated through the array of test data.

Here, the lowercase "i" represents each list within the array for 251 data points. So, by indexing each we're able to retrieve values for temperature, pressure etc for all 251 data points (i.e. if all the values for temperature were the first values for each data point, we could consequently append "i" at index 0 to the empty array "tempvalues", resulting in every value for temperature being added to a new list). The program repeats this for all the variables and creates an array for time values "tvs", which also has 251 data points.

3.3.3 GUI

By creating a Graphical User Interface using python – we are able to implement popular data science packages such as pandas and matplotlib to further analyse the data, create graphs, draw comparisons and much more, which will aid us in our secondary mission.

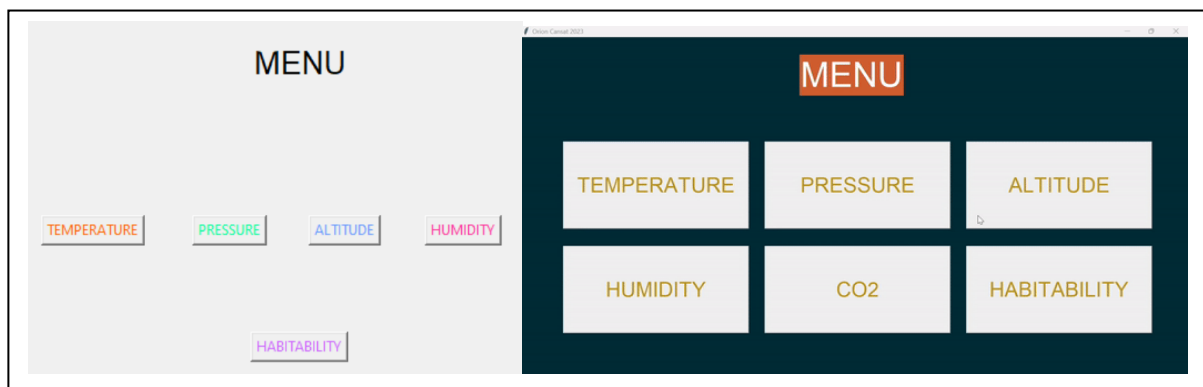
The GUI is built using the tkinter module and utilises different classes of frames (different windows), which have their own unique functions, to create pages which have elements such as buttons, graphs and labels. The graphs are created using data from a .csv file that is created at the end of the program which receives the data from the APC 220. The GUI runs as a



mainloop, which means it can carry out the actions of the different pages indefinitely (or until the program is shut down) so it functions in a similar manner to the original app.

To achieve a higher level of sophistication, a separate file for each group of functions is treated as a module and imported into the final program that executes the GUI. This is an improvement from the previous application as it allows changes to be made more efficiently and improves readability; prior to this change all the functions were defined and executed in one large program, making the application cumbersome and difficult to edit. Additionally, the final GUI has a much more advanced design for the menu page, benefiting from larger buttons which take up more space on the screen, improving the user experience.

The main menu consists of five buttons which lead to five different pages. The factors: 'Temperature', 'Pressure', 'Altitude' and 'Humidity' all lead to a page displaying a final animated graph constructed using the data from the CanSat's journey as explained in Section 3.3.1.

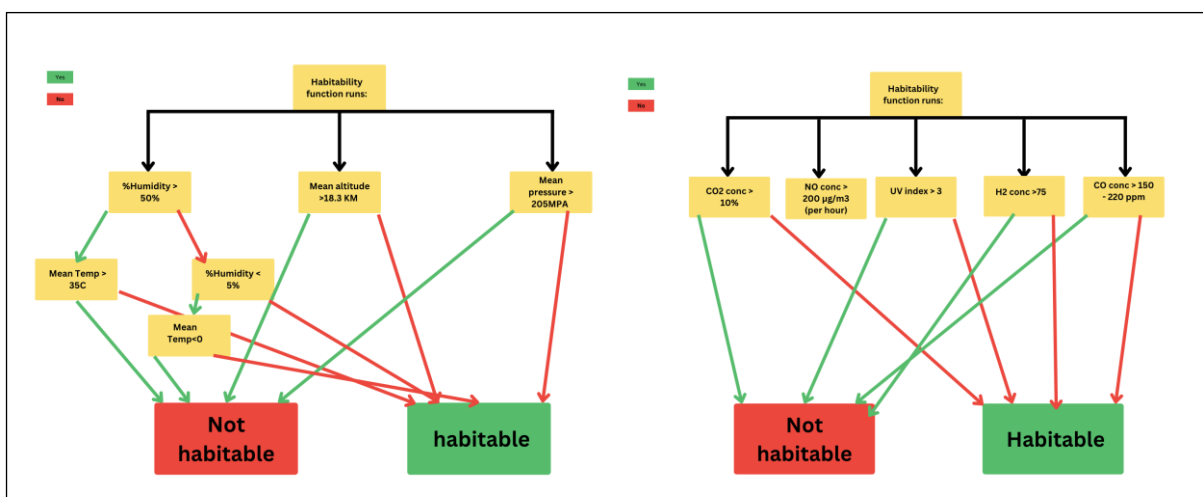


See above: the old GUI (left) and the new GUI with animated graphs using data from our regional launch (right)

3.3.4 Habitability

For our secondary mission we had to create a program that would analyse the data received from the CanSat and determine whether, given those conditions, the exoplanet our CanSat would be on is habitable for humans or not.

Below are 2 flow diagrams for the function "Habitability Check" On the left is the check for the P.A.T.H. sensors and on the right is the check for the secondary mission sensors. The Habitability Check then only returns whether the planet is habitable if both the P.A.T.H. and secondary mission sensors say that the planet is habitable. See Appendix 3.1 for the code.



After further research the values for our chosen parameters were changed, this improved the program as the habitability check previously returned uninhabitable for conditions humans would be able to survive on. By amending the parameters, the program is now more accurate. Above 6.1km humans require supplemental oxygen in order to survive⁵, without which the body suffers from hypoxia which impacts the ability to carry out homeostasis, eventually leading to death. Once the Armstrong Limit (approx. 18.3km)⁶ is reached, human life is considered unsustainable, constituting the parameter for altitude. The average human bone can withstand a compression of 205MPa⁷ along its length, and so this is the parameter we have chosen for pressure. If exposed to carbon dioxide concentrations greater than 10%⁸ (our parameter value), most humans will suffer from carbon dioxide poisoning. Symptoms include comas, convulsions, cardiac arrhythmias and death. If the carbon dioxide is solid (dry ice) burns are also common. With the ever-imposing impacts of rising CO₂ levels on global warming and climate change, we have also decided to consider how the concentration of CO₂ levels impacts the sustainability of human life in the long term, as detailed in the further data analysis section. Above 150-200 ppm carbon monoxide⁹ becomes life-threatening, ultraviolet radiation can cause (fatal) skin cancer at UV indexes above 3¹⁰ and hydrogen is flammable at concentrations between 4% and 75%¹¹. Therefore anything on the higher end would burn any probes, rockets or people on board. If the nitrogen oxide levels exceed 200 µgm⁻³ ¹² per hour it is deemed dangerous by the UK government: all the figures contribute towards the habitability parameters. While short bursts of exposure to high concentrations / intensities of the conditions mentioned above is dangerous – this is only the first stage to determining habitability. Once the exoplanet passes these initial checks, we then need to review how these conditions change over time, which is achieved by the python program. The effects of high altitudes may only be evident after many years of living at a higher altitude than the body is accustomed to. These effects could result in a change in bone density and lead to osteoporosis. Similarly, years of exposure to gases such as nitrogen oxide lead to respiratory diseases.

Humans can survive at a range of temperatures and humidities, however it is generally accepted that prolonged exposure to extremely high humidities and temperatures is unsustainable for human life as the body is unable to regulate temperature and will likely suffer from hyperthermia and/or heat stroke. At extremely low temperatures most humans will suffer from hypothermia without adequate equipment, however it is possible to suffer from this condition at relatively higher temperatures (just a few degrees below 0 Celsius) if unaccustomed to the climate. When the humidity is too low, all surfaces on the human body exposed to air won't be able to function correctly as this interferes with homeostasis. Since temperature and humidity are dependant, if the temperature is greater than 35 degrees C and

⁵ <https://biology.stackexchange.com/questions/42436/what-is-the-maximum-altitude-where-humans-can-survive>

⁶ <https://education.nationalgeographic.org/resource/elevation/> / <https://biology.stackexchange.com/questions/42436/what-is-the-maximum-altitude-where-humans-can-survive>

⁷ [https://phys.libretexts.org/Bookshelves/Conceptual_Physics/Book%3A_Body_Physics_-_Motion_to_Metabolism_\(Davis\)/07%3A_Strength_and_Elasticity_of_the_Body/7.01%3A_Strength_of_Human_Bones](https://phys.libretexts.org/Bookshelves/Conceptual_Physics/Book%3A_Body_Physics_-_Motion_to_Metabolism_(Davis)/07%3A_Strength_and_Elasticity_of_the_Body/7.01%3A_Strength_of_Human_Bones)

⁸ <https://pubmed.ncbi.nlm.nih.gov/16499405/>

⁹ <https://www.cpsc.gov/Safety-Education/Safety-Education-Centers/Carbon-Monoxide-Information-Center/Carbon-Monoxide-Questions-and-Answers>

¹⁰ <https://www.cancerresearchuk.org/about-cancer/causes-of-cancer/sun-uv-and-cancer/the-uv-index-and-sunburn-risk>

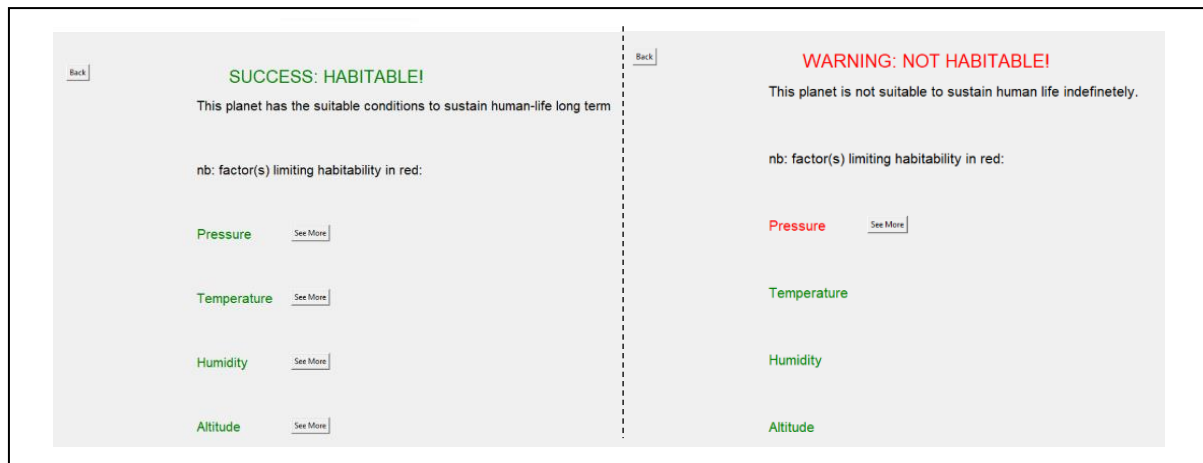
¹¹ <https://h2tools.org/bestpractices/hydrogen-flames>

¹² <https://www.gov.uk/government/statistics/air-quality-statistics/nitrogen-dioxide>

the percentage humidity is greater than 50% the planet is not habitable; alternatively, if the humidity is less than 5% and the temperature is less than 0 degrees C – the planet isn't habitable.

If the outcome is “not habitable”, the program returns “Human life would not be able to survive indefinitely on this planet” and presents the conditions which caused the inhabitability.

These outputs are displayed in the GUI so that the user has all of the information available to them in one place. The button ‘Habitability’ on the Menu Screen triggers the program to perform the Habitability Check function. Then one of the 2 screens shown below is displayed:

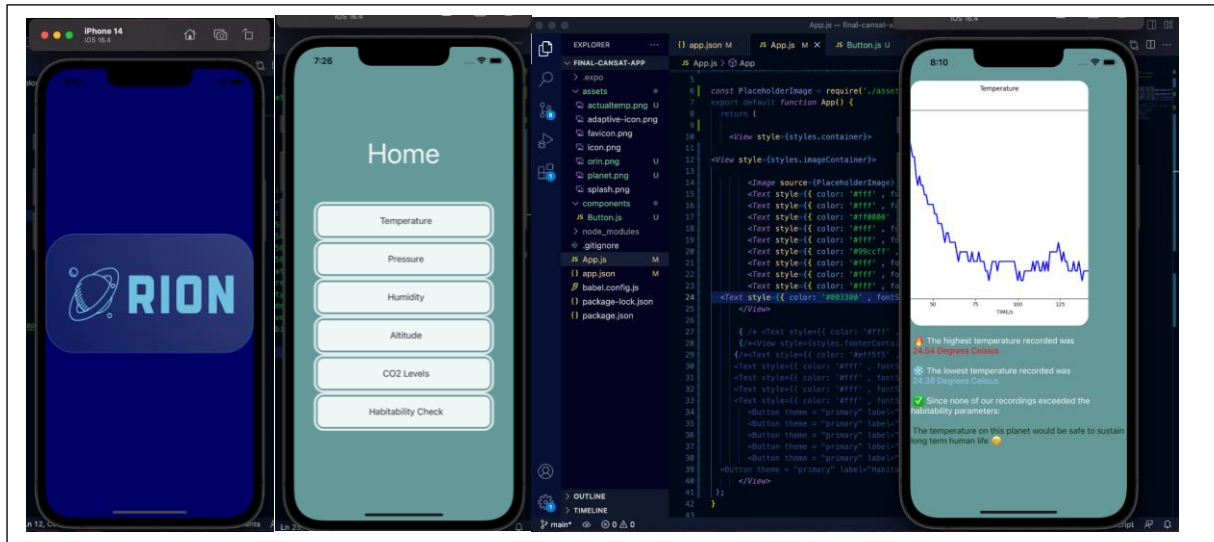


The “See More” button navigates to a page which displays a bar chart (created using matplotlib) comparing our values with the averages on Earth as well as Mars and the parameters decided earlier.

3.3.5 App

Furthermore, to improve accessibility a mobile friendly application was developed using react native and JavaScript. The application can run on all platforms. This has the added benefit of being easier to download and use for the average person. The general functionality of the app is similar to the GUI, however the graphs must be uploaded to the folder manually in order to be viewed on the app. The app was built by creating separate JavaScript files for each component, for example the buttons, and then importing those into a final file which executes these.

In addition, the app allows users to save the graphs directly into their camera role or share them on social media. Complete with a splash screen, home screen icon and in-depth descriptions of each graph, the app is an interactive and comprehensive means of engaging with the general public, helping to generate interest in data analysis. “In progress” is the comment box feature which will allow users to leave comments about the results. This could act as a way for members to communicate with each other as the probe roams the exoplanet and allows ground control to stay up to date with the environmental conditions on the planet.



3.3.6 On-board software

To create the on-board software we had to test that the sensors were recognised on the I2C bus for perfboard A. Using a widely-used I2C scanner online, perfboard A was connected to an Arduino Uno and the I2C code uploaded, this yielded the result that all 3 sensors were detected on the I2C bus, each with unique addresses. This meant that an I2C multiplexer was not needed, as it is only needed when multiple I2C devices have the same address. This ultimately saved time and money.

Through a process of elimination, having previously found that the I2C address of the MICS6814 sensor was 0x19 and that the I2C address of the SCD41 sensor was 0x62, we deduced that the I2C address of the LTR390 UV sensor was 0x53. We then went on to test perfboard B to see if both the MICS-2714 sensor and the GPS module could work simultaneously, however when the MICS2714 test code and Adafruit GPS Echotest code was run, no result appeared on the serial port monitor.

After realising that sometimes the use of the TX RX pins by the GPS module on the Arduino Uno interferes with the analogue readings coming in from the MICS-2714 sensor, we then disconnected the GPS module from the circuit and retried the test using the MICS-2714 sensor alone, which solved the issue.

As a result it seems likely that the Adafruit GPS module that we had originally planned to reuse from a previous year's Cansat project to save money and resources, would need to be replaced by an I2C breakout sensor such as the PIM525 GPS breakout, so that it can work alongside the other electronics, without interrupting the signal by using the TX RX pins, instead opting for use of the A5 A4 pins.

The onboard software was updated to accommodate for the addition of all the new sensors, and the MICS2714 sensor commented out in order to test perfboard A and the BME280 sensor in isolation. Attached is the updated code, which contains several amendments used to test and debug the individual sensors as we ran into several issues (as in the previous review, a full version of the code has been made available on our GitHub¹³).

We then tested the software running on perfboard A and ran into a few issues. The first was that whilst all the sensors on perfboard A were recognised on the I2C bus, when the CanSat

¹³ https://github.com/IncJean/Cansat-2022-23/blob/6a285d98cb37c130e2bca9debc32b32e3541167e/Cansat_alpha_code_v2

software code was run (See Appendix 3.2), nothing would display on the serial port monitor. Evidently, whilst the wiring of the electronics was correct and the program itself had been written without error and was able to compile within the IDE, there was some other underlying issue at hand. After scouring forums and articles¹⁴ on using multiple I2C sensors with an Arduino Uno, we concluded that this could have been down to a hardware issue: namely, the absence of pull-up resistors in the circuitry as well as a potential power budget issue.

Pull up resistors of 10k or so were needed to ensure that the voltage didn't dip when the Arduino was connected to a power supply.

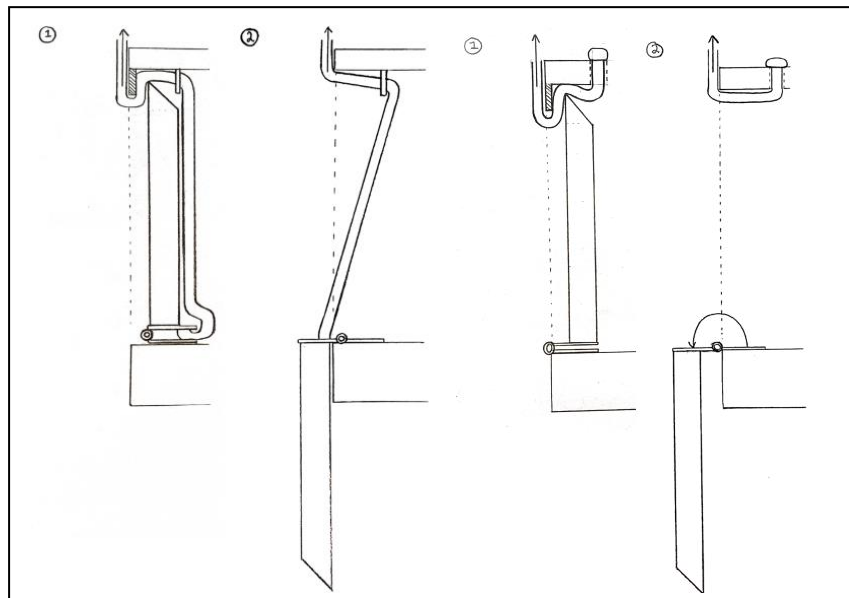
3.4 Landing and recovery system

In order for an effective landing we decided to have both a parachute and legs attached to our can.

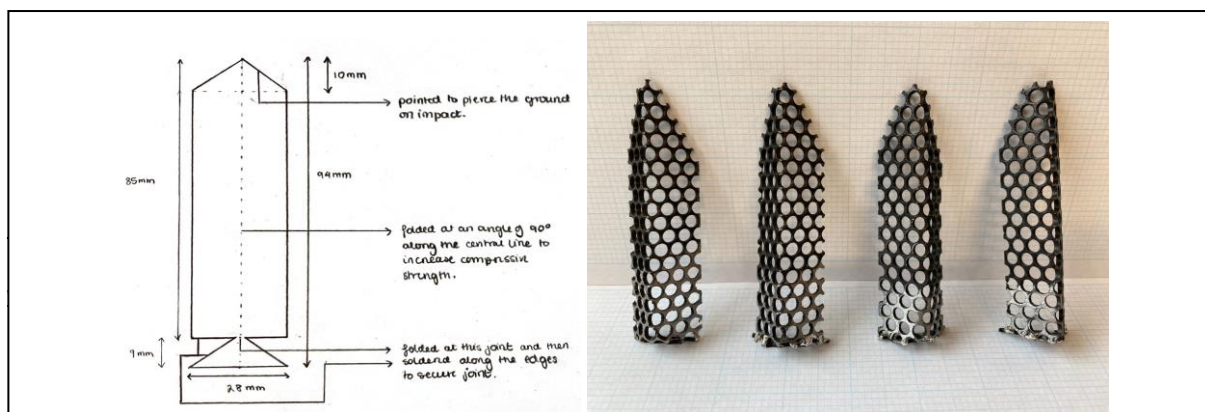
3.4.1 Initial leg design

Having compared the properties of aluminium, steel and copper for leg material, we came to the conclusion that although aluminium would be the ideal choice because of its high strength to mass ratio, it would not be a feasible option as it can be neither soldered nor welded - which is a requirement of the design. Therefore, we concluded that steel would be the best choice, as it is more resistant to wear than copper. Furthermore, we decided to use a punctured steel to reduce the overall added mass, effectively making it a stronger and lighter option than aluminium. In order to ensure the legs would not snap on impact, given the introduction of holes into the material, we decided to crease the steel: increasing its compressive strength. In the PDR we had decided that the legs will be motor controlled. We then edited that to the mechanical design with the parachute chords and then finally onto the sprung hinge design. See below for the change in design of the legs:

Iteration 2 (left) of landing system relied solely on the upward force of air resistance on the parachute to remain in the landing position. This is remedied in the final design (right), where a sprung hinge will keep the legs under tension in the upright position, so that they will remain in the landing position when released.



See below for the leg design from the CDR:



After the regional launch we made various changes to the leg landing system:

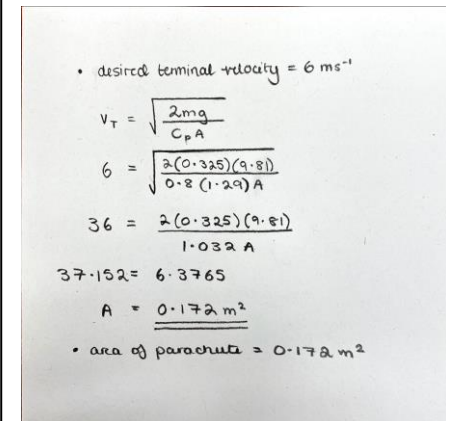
PREVIOUS DESIGN	CHANGES MADE SINCE CDR	REASON
4 legs	3 legs	Offers more space for electronics when legs are in the upright position
Punctured steal	Corrugated aluminium	Aluminium is lighter and stronger
Fold requiring soldering in leg	Fold requiring no soldering in leg	The soldered joints of the previous legs were too weak, and snapped under the pressure of being in an upright position
Lip to hold leg upright	No lip	Upon consultation with launch team at local launch we realised there was no need to have a lip to hold the legs upright, because if they were rounded at the top, they could be loaded into the rocket and the force of propulsion would be enough to release the can without the legs catching
Pointed tip of leg	Rounded tip of leg	Prevents catching on side of rocket during launch
Wide legs	Thinner legs	Increases strength of can walls as apertures are smaller
Long parachute chords	Shortened parachute chords	During the local launch, the parachute did not deploy as the chords were too long and therefore became tangled. Despite bulleting down, our can and all are electronics remained functional and intact
No groove in crumple zone	Groove in crumple zone for plasticine	Adding plasticine improves the function of the crumple zone by increasing the compressibility of the design

3.4.2 Parachute

We started the parachute research with some calculations. Firstly, we calculated the descent rate of the can without a parachute simply based off of the contact surface area of the can with air resistance. We used a drag coefficient of 0.8 which is approximately equivalent for all cylindrical type shapes. Then using the equation for terminal velocity we calculated the terminal velocity to be 42.5 ms^{-1} . Additionally we calculated the force of drag on the object still without parachute using the drag equation. It came out to be 3.188N.

The desired rate of descent for our satellite is 6 ms^{-1} since we are attempting to land the can on legs. Therefore, with the new calculated information and the desired terminal velocity, we calculated the required area of a parachute to allow our can to descend safely and in a controlled manner. The resulting area was 0.172 m^2 . We then double checked the calculations by making sure the drag matched up with the falling force.

Finally, knowing the area, we worked out the needed radius of the parachute, $r = 0.2342 \text{ m}$. We then tested our parachute (further details in Section 3.6.2) and, due to advice from OneWeb, decided to buy a parachute instead.

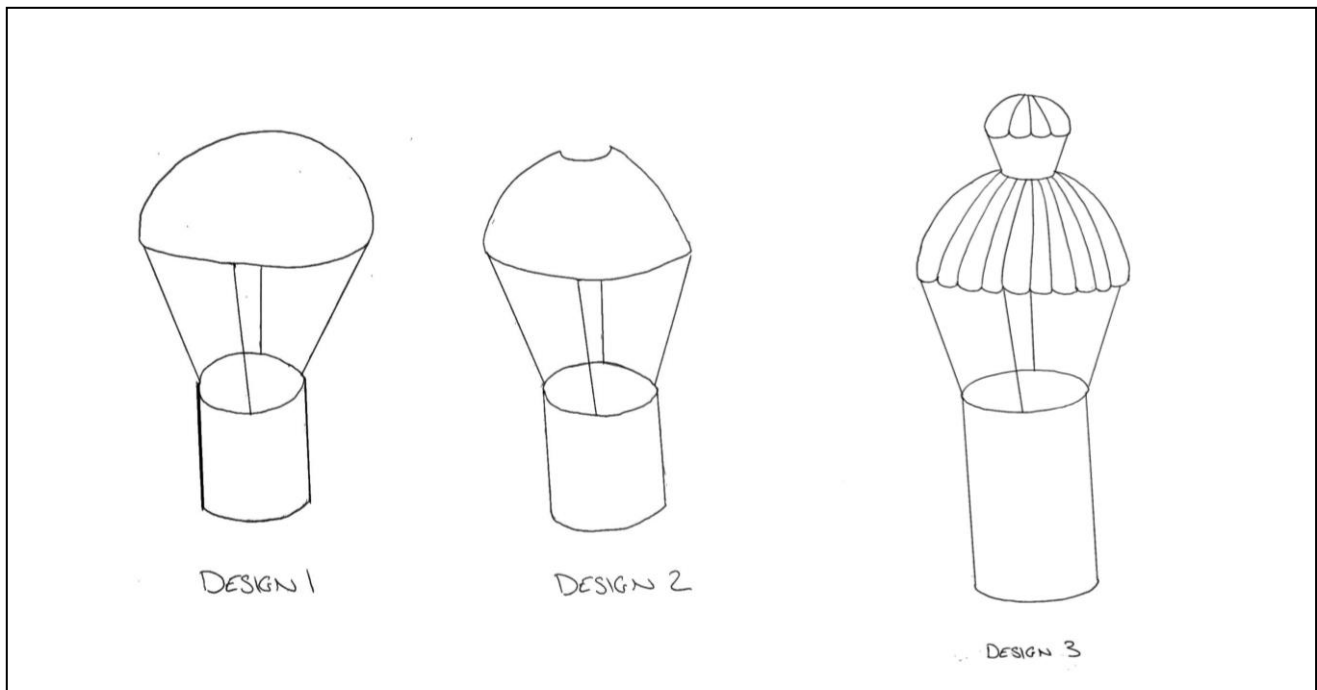


Handwritten calculations for parachute area:

- desired terminal velocity = 6 ms^{-1}
- $$v_t = \sqrt{\frac{2mg}{C_d A}}$$
- $$6 = \sqrt{\frac{2(0.325)(9.81)}{0.8(1.29)A}}$$
- $$36 = \frac{2(0.325)(9.81)}{1.032 A}$$
- $$37.152 = 6.3765$$
- $$A = \frac{0.172 \text{ m}^2}{1}$$
- area of parachute = 0.172 m^2

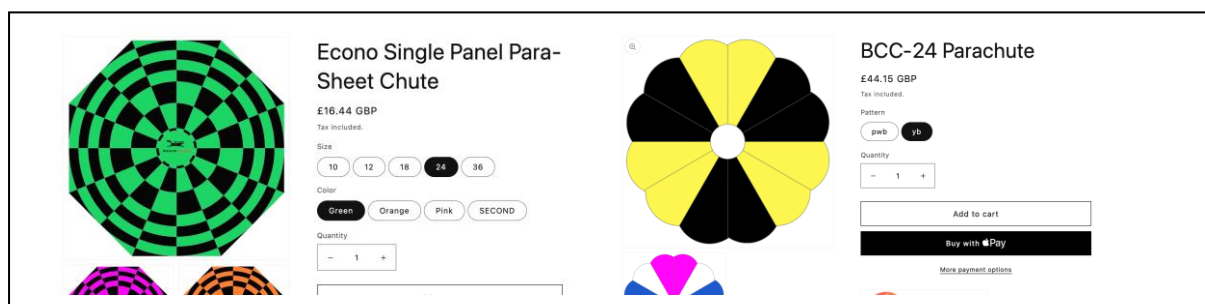
Buying a pre-made parachute meant we had much more scope for different designs, so first had to research what we thought would be the best one. Research has shown that the traditional rounded top parachute is quite effective and used in model rocketry. It does have some side effects however with a swing effect caused by air escaping out from the bottom of the chute. This is shown in Design 1. Alternatively, Design 2 involves a circular opening at the top of the chute allowing airflow, which will significantly reduce the swinging effect and will help stabilize the satellite for direct landing on the ground. Finally, the most elaborate and widely used design for rocketry is Design 3. It includes most of Design 2 with the added bonus

of having a pilot chute attached on top to allow an instant deployment of the main chute. This would ensure that the parachute does inflate correctly and also make up for the lost surface area as a result of the opening.



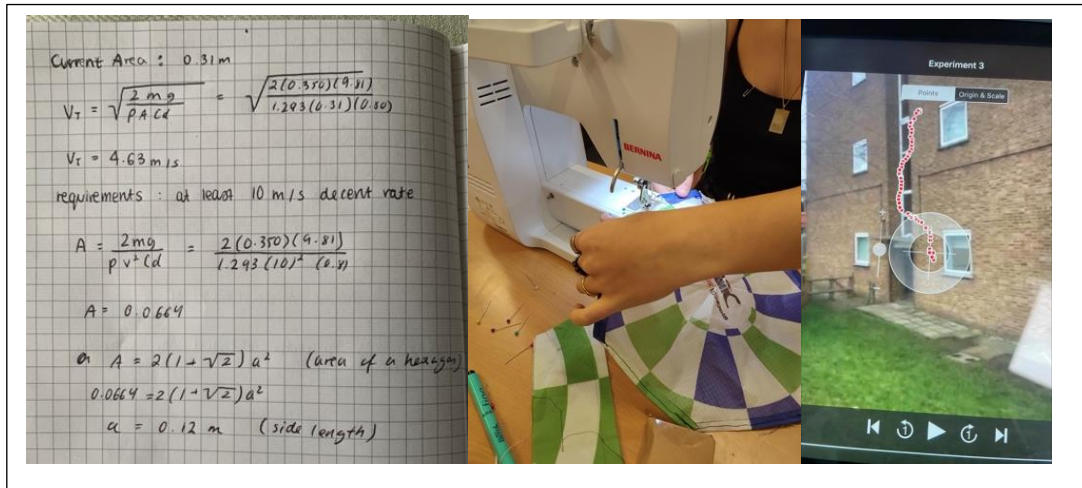
For our parachute, we decided to buy it from Black Cat Rocketry since they are professionally made and sewn. This will ensure that the bindings are secure to both the can and the parachute. Additionally, they come in a variety of vibrant colours which will help us locate the Can during descent and recovery.

We chose to test two different parachute designs as we finalised our Can design. The first is the Econo Single Panel Para-sheet Chute with a diameter of 24 inches. Which is slightly larger than we calculated to be but it leaves a larger margin for unexpected forces which could affect the Can's decent rate. It resembles Design 1 as it is a hemispherical dome that is attached to MIL-Spec para-cord which passes military grade parachute specifications. This parachute will be tested against the BCC-24 Parachute to see if the design actually influences the decent rate and which one is better adapted to our design. The BCC-24 is a 12-gore dome with an opening to allow airflow to reduce swinging. This design resembles Design 2/ Design 3. The surface area is slightly larger due to the gores and therefore may reduce the rate of descent even more.



After having conducted our own testing, we handed our two parachutes to a physics class so that they could run their own experiments. Using a tracking technology and attaching a weight to the end of the parachutes, they concluded that the final velocity of the parachutes was around 2 ms^{-1} . Now this isn't entirely representative of our case since the weight was

equivalent to the minimum mass of the can but nevertheless it was too slow and did not fit into our requirements for the competition. Using the terminal velocity equation and setting the final decent rate to 10ms^{-1} (as required in the guidelines), we calculated that the area of parachute needed to be 0.0664m^2 . Using a sewing machine, we reduced the parachutes sides to 0.12m and resowed the edges. We opted for a hexagonal shape as it was easier to sew and add on any hemming to ensure the structural stability of the parachute's sides.

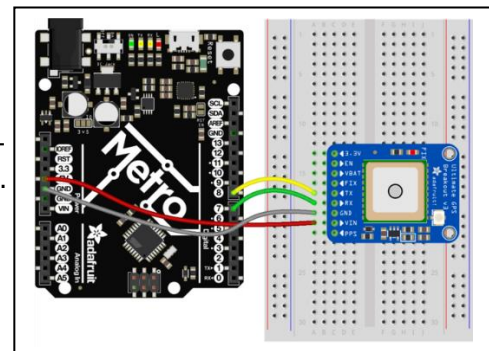


After our regional launch in which the parachute didn't open we decided to shorten the strings connecting the Can to the parachute. We also realised that relying on a lip to snap off for the legs to deploy was too unreliable, so after talking to the rocket engineer onsite concluded that it would be best to simply fold the legs upright before putting them in the rocket so that they deploy immediately on separation from the rocket.

3.4.3 Recovery system electronics

The initial plan with the recovery system electronics was to reuse the Adafruit v3 Ultimate GPS breakout module, that a previous year's cansat team had used – in order to cut costs and reduce resource consumption.

The wiring diagram on the right, courtesy of Adafruit¹⁵, details how the GPS module is connected to the Arduino:



We then attached this to the Arduino alongside the MICS2714 module as they were soldered to perfboard B. Software-wise, we then ran Adafruit's test Arduino code named 'GPS_HardwareSerial_EchoTest' on the Arduino Adafruit GPS Library, to test that it gave valid values, and when run, whilst it sent values to the serial port monitor, it included void data, which evidenced, alongside the absence of data when run alongside the MICS2714 sensor, that the use of the RX TX pins by the GPS module was causing an issue with the software.

A decision was thus made to switch to the PIM525 breakout GPS sensor in order to use the I2C bus instead and avoid using RX TX pins.

¹⁵ <https://learn.adafruit.com/adafruit-ultimate-gps/arduino-wiring?view=all>

3.5 Ground support equipment

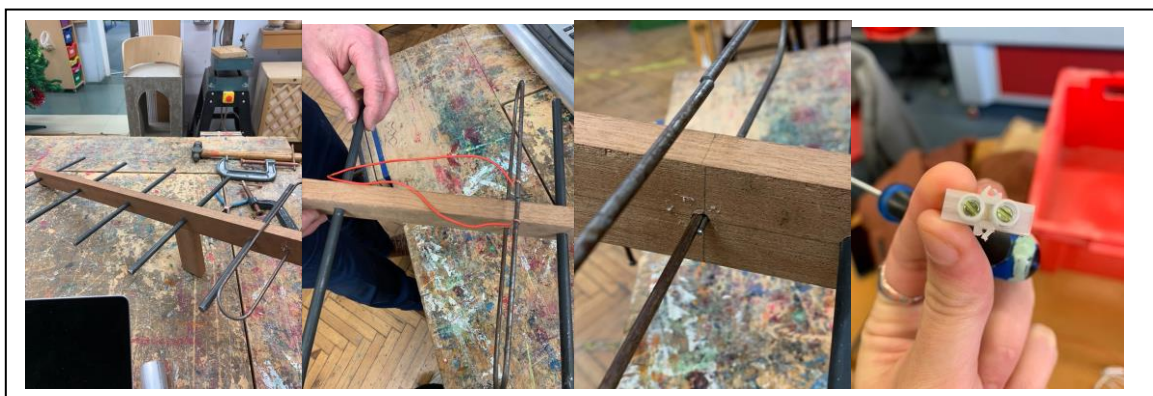
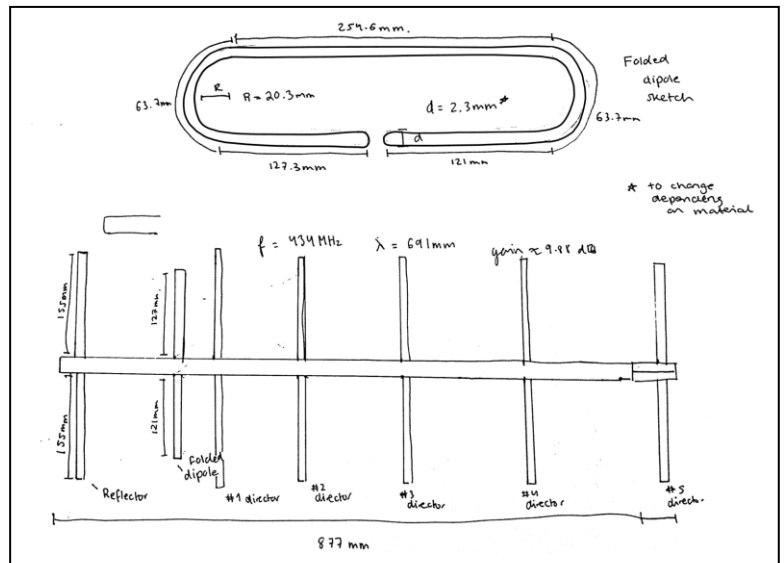
Our ground support station will consist of a Yagi antenna and a laptop to receive the signal from the Yagi with a wire. We will also bring spare laptops with us (under the advice from the OneWeb engineers) in case something goes wrong.

We will use a Yagi antenna in order to amplify the signal coming from the satellite's radio antenna. Using the value that I.J.J. provided for the frequency of the radio signal, we inputted to the YAGI calculator. The program then outputted the correct measurements for a frequency of 434MHz and a gain of approximately 9.88 dBd. The design

includes 7 elements: a reflector, a dipole, 5 directors, and the boom. For the dipole we decided to use a folded one instead of straight one because it has a higher input impedance. There will be a gap left where we can solder a wire that will then send the amplified signal to the receiving computer via a wire. We will be using stainless steel as the material for the directors and dipole due to its high conductivity and sturdiness. For the boom we used PEP wood to insulate the dipole and directors.

To build the Yagi, we used a hack saw to cut to the stainless steel 10mm rod into the specific lengths. Once all the pieces were acquired, we marked on the 940mm piece of wood the centre line of the boom. To ensure all the elements would be in line, we placed a cross between the centre line and the position stated in the YAGI calculator. Using a pillar drill, we cut the piece of wood to ensure the holes made would be in line with one another. To fit the 10mm directors and reflector, we bored a hole exactly 10mm and hammered the rod into the boom. This ensured they would stay in place.

For the dipole, we curved the 23mm thick rod around an approximately 40mm curved surface to create the desired shape. Issues arose when we bored the whole too wide for the dipole making it loose fitting in the boom. To fix this we added a small notch and a hammered nail to wedge the dipole in place.



To ensure a secure connection between the wire and the Yagi antenna, we decided to add a chocolate block connector. This will ensure the link between the Yagi amplified radio transmission from the can and the receiving computer is strong.

3.6 Testing

3.6.1 Mechanical testing

The mechanical testing consisted of 2 main tests: testing the mechanism of the landing system and the compression of the walls.

The landing system mechanism was tested in order to check the effectiveness of the leg system when the parachute opens. This was done by recording whether the lip snaps when the force of the parachute is applied.

Lip length /mm	Iteration 1	Iteration 2
4	Snapped when released	Snapped before parachute release
6	Snapped when released	Snapped before parachute release
8	Snapped when released	Snapped when released

In conclusion, the 8mm lip length will be used in the final design, as the weaker options snapped prematurely due to the force applied to them by the sprung hinge.

We also tested how the 3 different designs (in Section 3.1.2) of the 3D printed walls survived compression by applying force to each using a table clamp.

	Design 1	Design 2	Design 3
Fracture?	No fractures	Completely snapped horizontally	Hairline fractures along seams of hexagons

In conclusion, although design 1 seems to have the most compressive strength, the hairline fractures along the seams of the hexagons in design 3 prevent more pervasive damage throughout the entire wall. This allows the can to stay intact upon impact, as seen in the local launch, and therefore Design 3 will be used in our final can.

3.6.2 Electronics testing

All of the individual sensors were tested upon wiring, as detailed in Section 3.2.2.2

3.6.2.1 Electronics orientation testing

We decided to test how well the electronics withstand the force of impact in different orientations by dropping a soldered piece of steel at different positions.

	Horizontal orientation	Vertical orientation
State of solder	7/10 soldered areas fractured	3/10 soldered areas fractured

In conclusion, the vertical orientation of the electronics poses less risk to the soldered joints and terminals upon impact.

3.6.2.2 Testing and calibration of sensors

During our visit to OneWeb, we were advised to carefully calibrate our sensors, as sensors within our price range don't tend to be to the level of accuracy sensors used in professional scientific equipment are.

Sensors including the BME280 (pressure, altitude, temperature and humidity) and SCD41 (carbon dioxide), utilise a self-calibration algorithm that simply requires the user to continually read the values from the sensor in a relatively stable environment for around 20 minutes break-in time and 5 minutes calibration time respectively. In order to calibrate the MICS-2714 nitrogen dioxide and hydrogen sensor, Arduino code from their wiki page¹⁶ was used. Similarly

¹⁶ https://wiki.dfrobot.com/Fermion_MEMS_Gas_Sensor_MiCS-2714_SKU_SEN0441

for the MICS-6814, we utilised calibration code from an online Github repository¹⁷ which ran the sensor for 10 minutes and returned the baseline resistances.

The first practical step we took to calibrate the remaining sensors was to inquire at our chemistry department, whether there was a way of preparing a known ppm amount of a certain gas, using lab equipment that may measure nitrogen dioxide levels or carbon dioxide for instance. However, we were unable to do this as the nitrogen dioxide sensor and other equipment they had were no longer operational. Having said this, the department pointed us in the direction of MOT garages which may have nitrogen dioxide sensors, so we aim to do some final calibration by comparing values to the industrial nitrogen dioxide sensors in the coming week using the MICS-2714 and MICS6814 sensors, to identify systematic errors or similar.

In terms of the uncertainty of each sensor, from the datasheets provided with each sensor, the following uncertainties and ranges were given:

- MICS-2714: range: 1-1000ppm H₂, 0.05-10ppm NO₂⁵
- MICS-6814: n/a¹⁸
- SCD41: Uses photoacoustic sensor technology; high accuracy +/- (40ppm + 5%); range: 400 ppm – 5000ppm¹⁹
- BME280: Temperature - +/- 1.5 degrees Celsius; Humidity resolution: 0.008%RH with an absolute accuracy tolerance of +/- 3 %RH; Pressure: n/a²⁰
- LTR390: senses both ambient light and UVA, peak spectral response between 300-350nm

3.6.3 Software testing

On the software side we tested the GUI by using data collected from our school's competitors from last year's CanSat. Once the electronics for the primary mission was finished, I.J.J. gave the electronics to T.I. so she could then test the GUI using data collected at the same time. As a result, T.I. realised that the graph she was plotting was being created too quickly as the code didn't consider the rate at which the Arduino was transmitting the data because the program was previously only using an already complete .csv file. The testing of the on-board software is described in Section 3.3.5.

3.6.4 Landing system testing

3.6.4.1 Parachute testing

Given that we had some left-over parachute material from last year's CanSat project, we decided to begin creating and testing Design 1 of the parachute designs shown earlier in Section 3.4.2.

The first step was measuring and cutting the material to the desired radius of 0.2342m. We purposefully left an extra 0.01 m in case we wanted to sew in a seam or an extension. Upon inspection of the material, we realized it would be possible to punch a hole through it and secure a metal eyelet. This would provide a secure connection between the string and the parachute. For the chord we chose a nylon woven thread that felt slightly stretchy. This would ensure that when the string tensed up, it would be less likely to snap. We then attached the Design 1 parachute to a block of wood that weighed approximately 350g (our can's maximum possible weight) and went up the stairs to test it.

¹⁷ <https://github.com/eNBeWe/MiCS6814-I2C-Library/blob/master/examples/Calibrate/Calibrate.ino>

¹⁸ https://www.sgxsensortech.com/content/uploads/2015/02/1143_Datasheet-MiCS-6814-rev-8.pdf

¹⁹ https://cdn.shopify.com/s/files/1/0174/1800/files/Sensirion_CO2_Sensors_SCD4x_Datasheet.pdf?v=1642503159

²⁰ <https://cdn.shopify.com/s/files/1/0174/1800/files/bst-bme280-ds002.pdf?v=1662743150>

Our aim for this test was to see the block of wood's descent rate with the parachute as compared to without, as well as testing the structural integrity of our design and materials.

We did three tests of dropping our parachute and block of wood system at a height of approximately 7.3 m, and the average time of falling came out to be 1.48s. Then using suvat equations, we calculated that the block of wood's impact speed was about 9.86ms^{-1} . In comparison, only the block of wood dropped at a height of 7.3m fell in 1.04s. Therefore its final impact velocity was 14.0ms^{-1} . This shows that our parachute reduced the impact speed of the object significantly, meaning our design works well and inflates properly.

Problems arose with the swing of the parachute as it fell sideways which is not ideal if we are trying to land out can as upright as possible and with the connections between the string and the parachute. We had to retie the strings each trial meaning that the final impact velocity was definitely altered by the loosening of the string's connections. Additionally, the final descent velocity of 9.86ms^{-1} is not enough to ensure the safe landing of our can, especially if we are attempting to land it on legs. Therefore, by careful consideration and testing, we have decided to buy a parachute to ensure the safety of our satellite and try to give it a bigger chance of success.

$$\begin{aligned}\text{height} &= 7.3\text{m} \\ \text{time} &= 1.48\text{s} \\ \text{initial velocity} &= 0\text{ms}^{-1} \\ \text{final velocity} &= ?\end{aligned}$$

$$\begin{aligned}s &= \frac{1}{2}(u+v)t \\ 7.3 &= \frac{1}{2}(0+v) \times 1.48 \\ v &= 9.86486486\dots \\ v &= \underline{9.86\text{ms}^{-1}} \quad (3\text{sf})\end{aligned}$$

$$\begin{aligned}\text{height} &= 7.3\text{m} \\ \text{time} &= 1.04\text{s} \\ \text{initial velocity} &= 0\text{ms}^{-1} \\ \text{final velocity} &= ?\end{aligned}$$

$$\begin{aligned}s &= \frac{1}{2}(u+v)t \\ 7.3 &= \frac{1}{2}(0+v) \times 1.04 \\ v &= 14.0384615\dots \\ v &= \underline{14.0\text{ms}^{-1}} \quad (3\text{sf})\end{aligned}$$



When the parachutes arrived, we decided that the best course of action would be to test how they would handle being thrown off a building with a weight equivalent to that of our estimated can weight.

Parachute	Observations	Conclusions
BCC24	This design was thrown off with 450g weight at an angle. When thrown off, the parachute almost immediately righted itself and landed at the same time as the second design.	The opening at the top of the parachute does impact the landing angle positively. The airflow reduces the swing of the can and the strings themselves can withstand up to 500N, well beyond the value of the actual forces acting on it by the can.
Econo	When thrown off individually with an approximately 350g weight at an angle, the parachute landed on the	The strings attached can withstand 300N which is also adapted to our design but the increased swing and overall

	ground a significant angle which would not be ideal for our chosen landing position. There was a significant amount of swing.	smaller surface area due to not having any internal gores, makes this design less adapted to our needs.
--	---	---

We handed our two parachutes to a physics class so that they may run their own experiments. Using a tracking technology and attaching a weight to the end of the parachutes, they concluded that the final velocity of the parachutes was around 2ms^{-1} . Now this isn't entirely representative of our case since the weight was equivalent to the minimum mass of the can but nevertheless it was too slow and did not fit into our requirements for the competition. Too slow a fall rate and the can will be swept off by the breeze never to be seen again. We concluded that we would have to hem a parachute smaller so chose to do so with the Econo parachute as it was easier to cut and sew.

3.6.4.2 Impact zone testing

To test the structural integrity of the different impact zones we decided to do different tests.

The landing force test consisted of: placing the different impact zone iterations flat on a table; adding 100g mass plates to the top of the impact zones; keeping on adding 100g to each crumple zone until there is 1.1 kg of mass plates on each impact zone; take off the masses and check for any cracking, caving in or alterations in the impact zones; if the impact zone is clear then it has passed the test.

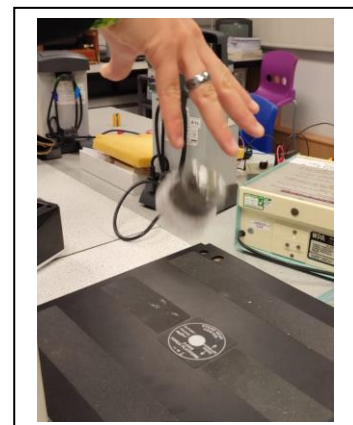
Impact zone	Mass added/kg										
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
Standard grid											
Gyroid lattice											
Diamond holes											

All of our impact zones sustained 1.1kg masses so we had to conduct more tests to determine which one was best.

The impact absorption tests consisted of 2 different experiments. 1 tested the different impact zones, the other tested different designs of plasticine added onto the end of each impact zone. The experiment is based on using the equation for the change in momentum overtime and a force detection plate to give us the force applied to the ground. We can therefore figure out how much of the impact force is absorbed over a specific amount of time. The longer the time, the less the force that could damage the internal components of the can.

Experiment 1:

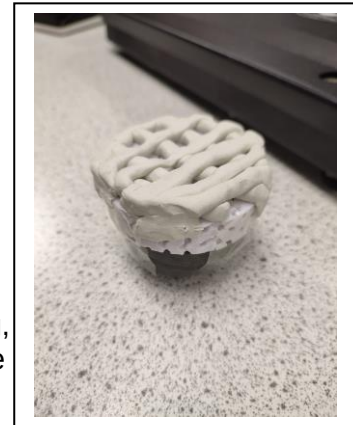
- We attached 350g of mass to the first impact zone using masking tape
- Then to ensure that each load was dropped at the same height, we held the bottom of the impact zone next to a tube
- Then by using the recording technology on the computer, we were able to record each drop's time and force on the force detection plate
- We then repeated the trial 3 times for each impact zone.



Experiment 2:

- We repeated Experiment 1 but this time by adding different designs made from plasticine to the gyroid lattice
- We kept the impact zone a control variable to just measure the influence of the different plasticine designs of the force absorbed
- We dropped each design and recorded the time/force on the computer.

The results (in Appendix) of the Experiment 1 showed that the gyroid lattice impact zone absorbed the most impact from landing, therefore best protecting the internal components of the Can. The results from Experiment 2 showed that the best design for reducing the impact force was the ring design.



3.7 Overall testing for launch

With our newly sewed parachute we decided to test it with our leg deployment mechanism and the strength of our 3D-printed walls. Essentially, the strings attached to the parachute will act as the trigger for the deployment of the legs. The force of the parachute opening should break the tabs that hold the legs in place, thereby forcing the sprung hinges to deploy the legs outwards. For our first test we attached the strings around the tab and through a hole in the lid, so it held the lid down to the can as well.

Once again, we threw our whole design of the roof of the science building and on its descent, the can hit the building's brick wall. This mean that when it landed, it was on its side and the can completely broke apart. On a positive note, the attachment of the parachute to the can held strong and the impact zone was in one piece. Unfortunately, we were unable to verify if the legs deployed at all since the design fell at a fast rate.

Conclusions from this test:

- Increase the infill of the can to make sure the walls can withstand a higher force.
- Rethink the deployment mechanism for the legs since the force of the parachute opening did not deliver enough force through the strings.



4 OUTREACH PROGRAMME

Having already presented out CanSat project to our Headteacher we are planning to present talks about our project to local primary schools in the area such as Alleyn's Junior School, JAPS and DPL. In order to make these talks more memorable and digestible for younger minds, we hope to host a stomp rocket competition (with edible prizes as an added incentive to participate!). On the topic of inspiring young people in STEM, we have hosted a number of talks and assemblies with the Upper and Middle School, explaining CanSat to them, and the skills you can learn as part of the project. This was very successful as we now inspired 4 teams to take part in CanSat next year! We are arranging whole school assemblies to help further our reach, and have collaborated with the Year 8 rocket club, introducing what CanSat is to them as well.

To get some advice on our CanSat we went on a trip OneWeb headquarters in London, on 07/12/22, which R.V then wrote an article about [here](#). By writing an article about it and posting it on the school website the message about our work spread further out into our school community. On the trip we were given an insightful tour of the offices, including the ground station control room and scale models of currently operating satellites. We then received a talk on the intricate inner workings of the satellite system, an introduction to what working at OneWeb would be like and finally were shown a promotional video on OneWeb's latest project. It was a truly inspiring opportunity for our CanSat team, as we presented our work about our CanSat to software and hardware to engineers there, most of whom had done a CanSat project of their own at university. We received constructive feedback on our project to help better the designs for our Can and our project as a whole.

We have decided to promote our brand on social media platforms (as the main demographic we aim to reach are teenagers) such as [Instagram](#) and have created a [website](#). We have given and are organising more talks and workshops in order to encourage and inspire more young minds to be interested in amazing STEM project opportunities such as this one. We use social media, especially our Instagram, to document our progress and spread interest to more people about our project. See below for what our Instagram page looks like.



So lovely to see you both on Friday. It was such a fun, energising session! And I was very impressed yesterday to see what the pupils put together for Instagram. They're already very good at communicating what they're doing in their science! There was so much discussion, I thought I'd put together a list of actions, just so we don't forget!

Actions from the meeting:

- We talked about measuring N02 where the Jags school street will be, first without the school street, and then later, once it is implemented. I think this would work brilliantly. Probably best to measure during the specific hours of the school street, as over a day the averages will show less of a spike. The dates of the implementation of the school street outside Jags aren't public yet, and it could be that it isn't until the summer. It would be good to measure it pre-school street, regardless. But in the meantime we could all have a think about another good place to do it. I'll have a think and ask around.
- Once you've got the monitors up and running, we've decided on a location, and you've done the measuring, we could work together to present the findings. I can help give you the context of air pollution, giving you information on the most up to date guidelines etc.
- Then we work out where to present the data!

Presentations:

- Fingers crossed timings work, but Mums For Lungs plan to hold an event with local MPs, councillors, and clean air advocates in June, around Clean Air Day. This would be a great opportunity for you to have a stall and discuss the science and your findings with key decision makers.
- Create an assembly for upper and lower schools, looking at data and impact of air pollution.
- Create a STEM workshop making simplified versions of monitors, with air pollution context, to give to Alleyns Juniors, as a prototype and see if we can do it in other local primary schools.
- Work with Eco Team (Ann Katrin) to disseminate. I could help an aspiring journalist to write a piece about it for local press?
- Work with Sustainability group (mix of students, teachers, and leadership team) to ensure the air pollution message is being heard by people who make decisions.

Phew! Lots of things to do. Do get back to me if I've misinterpreted anything, or need to flesh things out a bit more. Let's keep in touch and continue to work together.

Look forward to seeing you all again soon!

Claire McDonald,
Schools Lead
Mums For Lungs

Given that our secondary mission revolves mainly around the use of gas sensors we thought that we could use our can to help local organisation “Mums for Lungs”. Mums for Lungs campaign for better air quality around primary schools in London, however do not have any data that would be of specific use to them, so we have offered to collect data using the sensors in our Can. On different days and times we will walk/cycle round local schools to compare the different gas concentrations to help Mums for Lungs in their campaign. To more effectively do this and find out about what else we could do to help we met with the School Lead for Mums for Lungs, Claire McDonald. A summary of our meeting can be seen in the email above. Details about our work with Mums for Lungs will then be posted about in our school eco newsletter “Green Scene” to spread news about what we are doing to the larger school community. Mums For Lungs plan to hold an event with local MPs, councillors, and clean air advocates in June, around Clean Air Day. This is an opportunity for us to have a stall and discuss the science behind our findings with key decision makers such as the local MP Helen Hayes. Given that one of our members is a youth MP for Westminster we are also trying to get in touch with the Westminster MP to spread awareness about the dangers of air pollution.

We also took this opportunity to further educate ourselves and those in our school community. M.K. led a talk at Environment Society about air pollution, details of which will also be published in the Green Scene magazine. In the talk, M.K. described the different types of atmospheric pollutants, what causes them, the consequences of air pollution, the extent to which the world is polluted (99% of the world’s population live in areas where air pollution levels exceed WHO guideline limits!), discussed Ella-Kissy-Debrah’s death and subsequent legal case, and finally concluded with the work Mums for Lungs are doing to help reduce air pollution. Furthermore, M.K. is also participating in a Model United Nations conference on 04/03/2023 where in the Science-Fiction Committee (set in the year 2296), they will be debating the abandonment of Earth and relocation to an exoplanet, for which the work we are doing for the CanSat project will prove incredibly useful. Not only are we using the CanSat project to help others in other areas, it is helping us in other areas as well!

To boost morale and spread word, we were able to procure additional funding to buy CanSat hoodies (designed by R.V.) for the entire team. This helps us both represent the school and our project. In conclusion, we have taken many steps to organise and execute this project so far, especially with our outreach, and we plan to do even more in the future. We are all passionate about STEM and our project, and we hope to inspire others with our work.



5 LAUNCH DAY PREPERATION

5.1 Launch checklist/countdown and 5.2 Post mission checklist

	Task	Start time	End time	Max. duration /mins	Member(s) responsible	Other notes
	Livestream / post about events occurring	11:00	17:30	390	R.V.	Post also in previous days about launch, livestream launch itself perhaps take extra footage with drone
	Meet at school	11:00	11:15	15	All + Team Leader	Ensure everyone has packed lunch and is wearing CanSat hoodie
	Arrive at launch site	11:15	13:15	120	All + Team Leader	Make sure nothing is left in vehicle
	Allocated prep-table	13:15	13:30	15	All	Ensure prep-table has plug socket nearby
	Check Can for damage occurred in transit	13:30	13:35	5	C.L.	If damage has occurred, use back-up Can and/or hot-glue-gun broken section
	Check parachute for damage occurred in transit	13:30	13:35	5	E.C.	If damage has occurred, use back-up parachute
	Check electronics for damage occurred in transit	13:30	13:35	5	I.J.J.	If damage has occurred, use back-up electronics and/or resolder broken connections
	Set up main ground station laptop	13:30	13:40	10	T.I.	Check battery, keep spare laptop close by in case of emergency
	Set up Yagi antenna	13:30	13:35	5	E.C.	
	Test Yagi connection to laptop	13:35	13:40	5	E.C. + T.I.	
	Test sensors transmitting to laptop successfully	13:35	13:45	10	I.J.J. + T.I.	
	Buffer time	13:45	14:00	15	All	Spare 15 minutes incase anything goes wrong or tasks take longer than expected
	Safety Briefing	14:00	14:15	15	All + Team Leader	When going to safety briefing ensure everything left on prep-table is secured and protected
	Extra buffer time	14:15	14:30	15	All	
	Further launch prep	14:30	16:00	90	All	As detailed by Regional Launch Participant Guide
	Launch opportunity	15:00	16:15	85	All	As detailed by Regional Launch Participant Guide
	CanSat recovery	15:15	17:00	105	All	As detailed by Regional Launch Participant Guide
	Locate Can	(time of landing)	Unknown	Unknown	All	
	Save all data collected	(time of landing)	5 minutes after landing	5	T.I.	Store data both locally and on OneDrive, then share to chat so all members can access it
	Turn CanSat off	(time of location of CanSat)	5 minutes after location of CanSat	5	C.L. + E.C. + I.J.J.	
	Check all aspects of CanSat for damage	(time of location of CanSat)	5 minutes after location of CanSat	5	C.L. + E.C. + I.J.J.	
	Clean up	16:30	17:30	60	All	As detailed by Regional Launch Participant Guide
	Final social media post	17:30	17:35	5	R.V.	Prepared in advance

5.3 Launch Day risk log

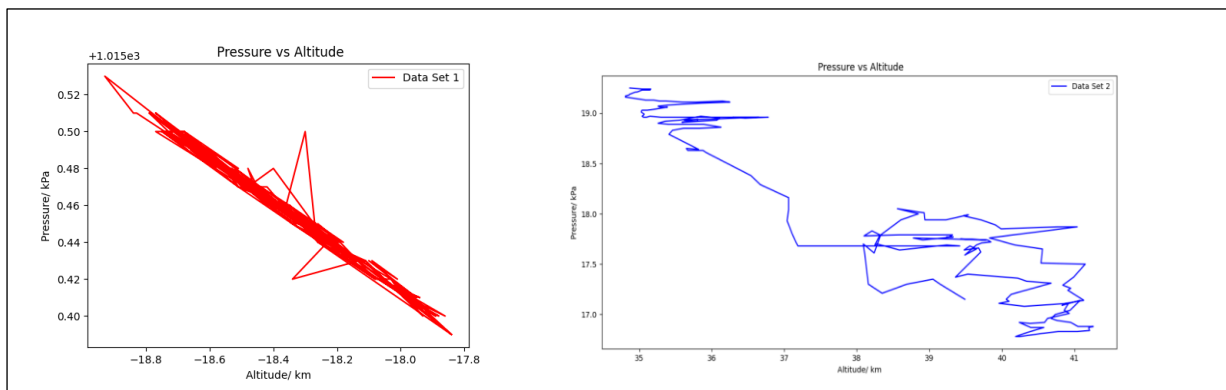
RISK	MITIGATION
Can / parachute gets damaged in transit	Bring spare Can, parachute, and legs, bring hot glue gun and soldering iron
Ground station doesn't work	Bring spare laptop with software already installed
Laptop runs out of battery	Bring spare laptop chargers
Team member is ill	Ensure everyone knows everyone else's role, Project Manager acts as spare person to cover
Transport vehicle breaks down	Long launch opportunity so we can still launch even if we arrive late, leave early from school for extra buffer time

5.4 Results analysis procedure

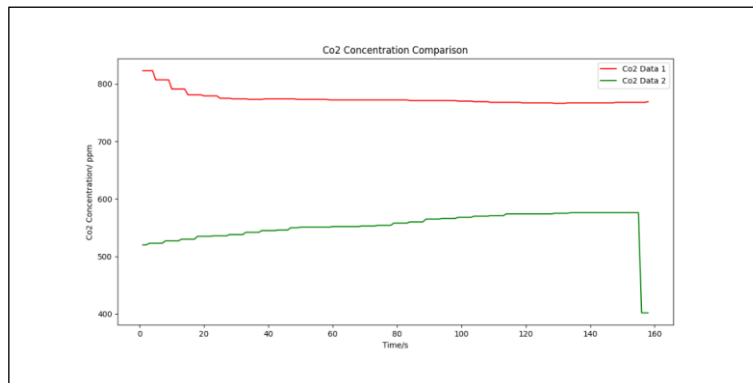
Our criteria for a successful mission is Having the cansat land safely, successfully receiving and recording data from the cansat, and having the data analysis and habitability check programs work. However, even if one or more of these things do not occur, the experience of doing the CanSat project as a whole is more than enough of a reward for our team.

Overall, we concluded that our launch was very successful, as the CanSat continually measured and transmitted data even from inside the rocket which was then recorded and turned into animated graphs. However, the parachute did not deploy due to the strings being too long so unfortunately the CanSat plummeted straight to the ground, yet despite this everything was still working when we found it! Although there were some hairline fractures around the hexagonal holes in the Can, thanks to the additional duct-taping the CanSat itself remained fully intact.

After the launch T.I. set about analysing the data. We had 2 datasets available to us, one that we took at school (Data Set 1) and the one from the launch (Data Set 2). As altitude increases, there are less air particles above you and so relative pressure decreases. This relationship is evident in the graph for Data Set 1, as the general negative downward gradient represents an inversely proportional relationship. Contrarily, the graph for Data Set 2 has much lower correlation coefficient as the relationship is much harder to make out. However, a general trend of inverse proportionality can still be observed. The difference is due to the more sporadic motion of the can in Data Set 2, however the relationship remains evident despite the more random data. These graphs were plotted using python in order to explore the relationship between different parameters. Plotting two related variables at once is much more useful than only plotting a singular variable as was the case previously, as it creates the opportunity to think more dynamically about the data and how different parameters interact and impact each other.



Additionally, we utilised python to explore the difference between the CO_2 levels in the field and at our school. The graph shows us that the concentration of CO_2 is much higher at our school (Data set 1) than at the field (Data Set 2). This could be because since the field is located in a rural area, there are less cars and people and consequently less CO_2 emissions – potentially leading us to correlate higher air quality with rural area - we are sure mums for lungs would appreciate this data. Data analysis like this is incredibly crucial as it allows us to draw clear comparisons between data sets; when considering supporting life on our exoplanet we can treat each data set as a different planet. In this case, it would be more beneficial to choose the planet from Data Set 2 as it has much lower CO_2 concentrations and so the greenhouse effect on that planet would be much less extreme. The sharp decrease in gradient for Data Set 2 shows us the quick downwards trajectory of the can after the parachute failed to deploy.



6 LESSONS LEARNED

C.L: While designing the can, I learned the importance of the cyclical design process, because our design went through so many iterations that would not have been possible without testing and reviewing. I also found that discussing the design with team members helped me look at problems from a different perspective, and improved my communication skills. I think communication was in fact the biggest challenge, because I had to translate the designs to the team as well as several 3rd parties in presentations to our head and Mum's for Lungs representatives.

E.C: At the start of this competition, I was unsure of my role in the team, I didn't know how to code, unsure how to use 3D design software and I certainly wasn't the most organised. Somehow, the most valuable lesson I learnt from these past couple months is that in a team, even the smaller jobs add up to influence the success of the final design. Even if it wasn't the backbone of the operation, I aided our project by learning how to construct a Yagi antenna from scratch (valuable for communication), research, calculating and constructing our parachute (valuable for safe landing) and being the second opinion for any design idea (valuable for realism). I have learnt to independently use what I have learnt in mechanics class and convert it towards a real, tangible solution. This has been extremely rewarding and has inspired me to consider a career in engineering. All in all, this project has taught me communicate with a team and work together for what will definitely be successful launch.

I.J.J: This experience has been invaluable for me. From learning how to use KiCad to draw electrical schematics, to solidifying my understanding of how to utilise Arduino's I2C bus to connect multiple devices simultaneously; the entire process of design and testing has developed my skills as someone even with a keen interest in electronics. After countless late nights and frustrating dead-ends, it was perseverance and teamwork that allowed me to overcome the many issues I came across. It was also through close communication between the team that I could work so closely with C.L. to turn their can designs into 3D models and feedback on any issues by 3D printing prototypes, and likewise with T.I. to help resolve ground station software issues with my own understanding of Python 3 and by improving my own Arduino code. Whilst there are still more issues to iron out with the UV sensor and MICS814 gas sensor, I believe that the secondary mission in its current state sets out some of the most valuable parameters for habitability. Similarly, I feel proud of the functionality of the primary mission, which has been operational since November.

M.K: Personally, I am so glad that I have participated in the CanSat project, as not only have I learnt much more about working in and organising a team, but also about 3D printing, design, software and electronics too. The experience of being Project Manager meant that I got to help

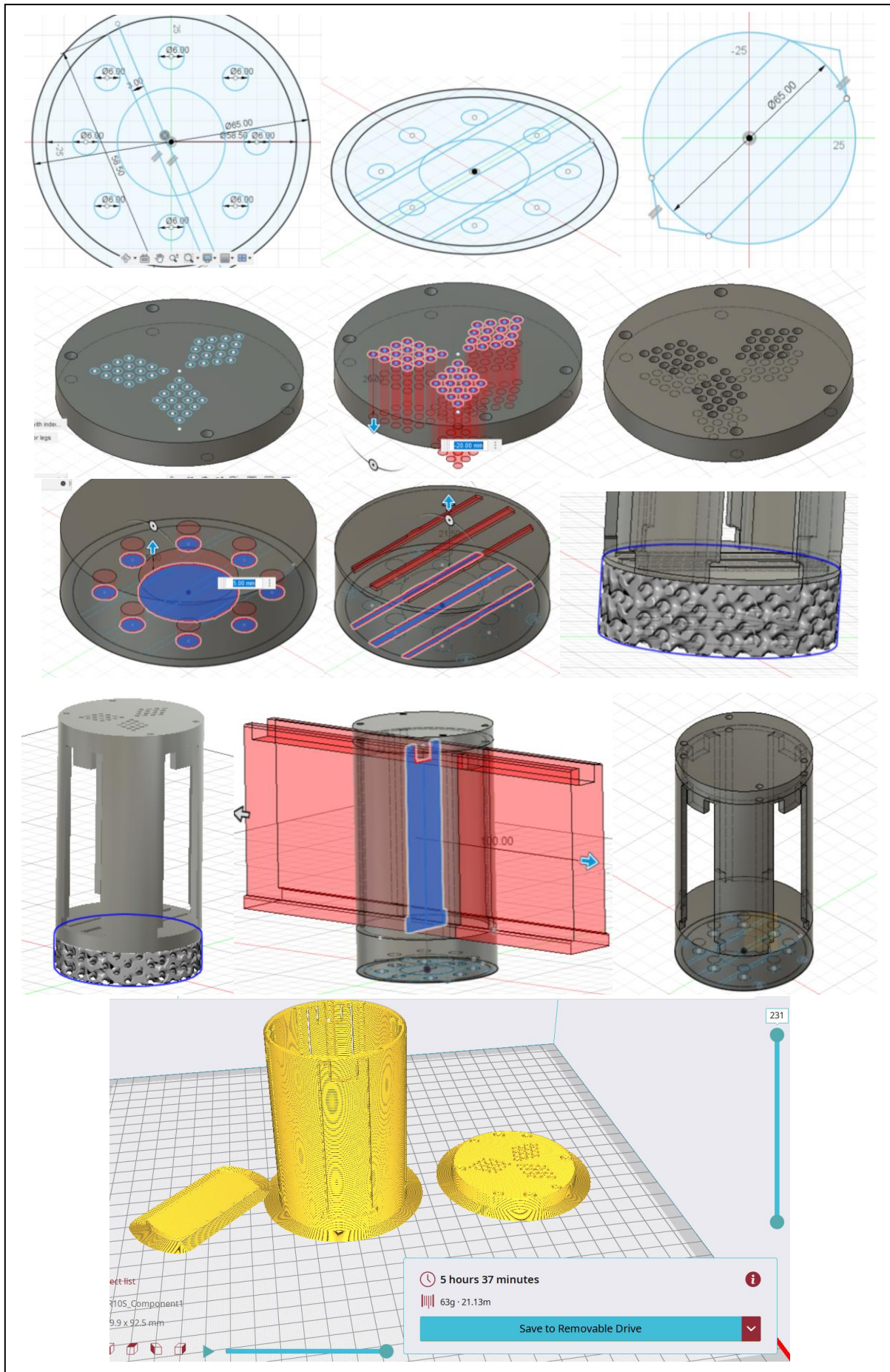
with all the different aspects of the CanSat project and has really benefitted me in other areas as well, such as being more confident when having to make another Gantt Chart for my CyberEPQ. I would like to study Computer Science, but prior to this project was unsure of where to go from there. Now I have a clearer idea of what it would be like to lead a tech project and after our visit to OneWeb am considering combining my interest in ComSci and tech with Astronomy into a career. The hardest part of the project was definitely the time management and ensuring that all team members were communicating with each other and working collaboratively. My favourite part was simply working together with others to create something and learning invaluable new skills along the way.

R.V: I have developed and gained new skills from this competition such as time management, teamwork, pitching projects, and budgeting/accounting. Through being Head of Marketing, I found a passion for advertising, and hope to use the skills I cultivated in this competition in the future. I aspire to have a finance or marketing career due to my involvement in this project. I am now partaking in an advertising course at my school to further this goal. The hardest part for me was time management but as a result, I became more organised. In the beginning, with all the extracurriculars and homework, deadlines were hard. Animating the Instagram posts was very time-consuming but I had fun expressing my creativity in marketing. The most enjoyable thing for me was pitching our project to Mums for lungs and the talks we hosted. These helped drastically improve my public speaking skills, and I made many connections with the STEM community at my school.

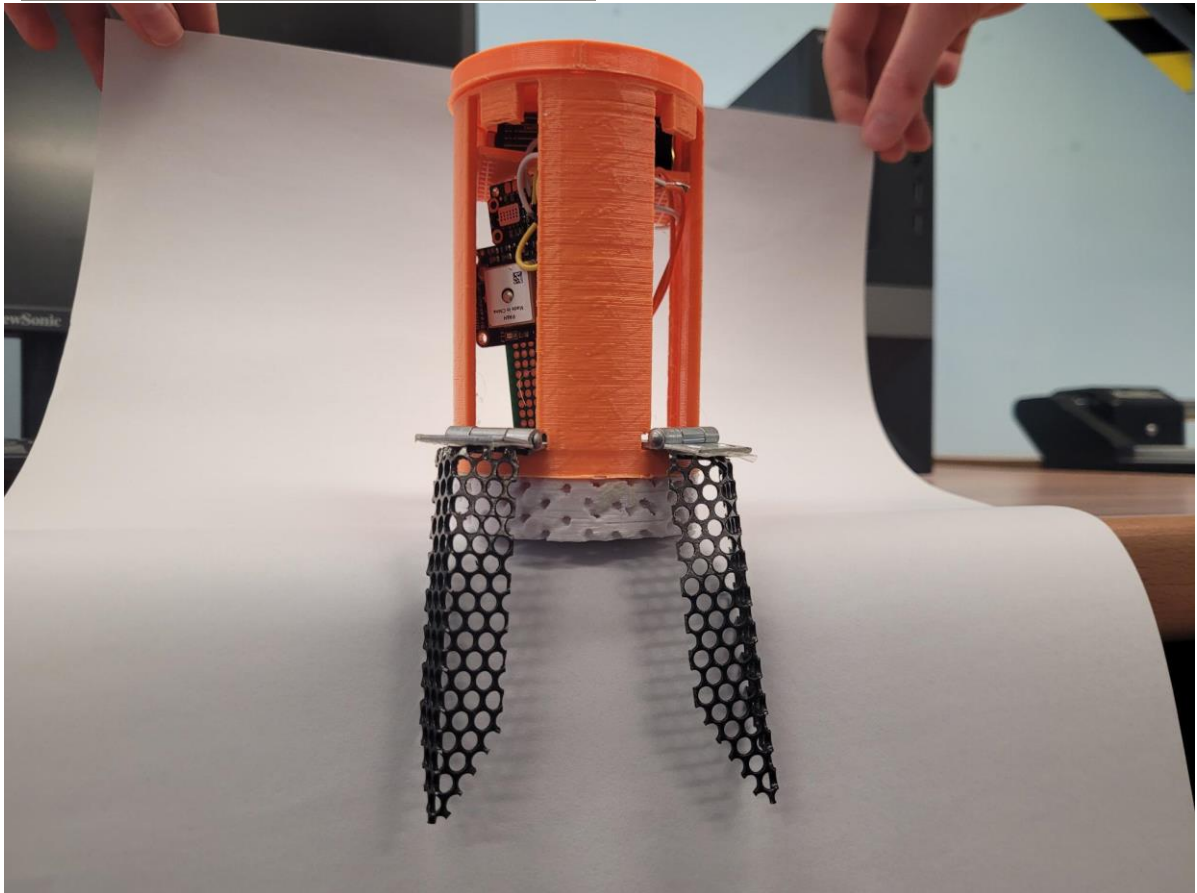
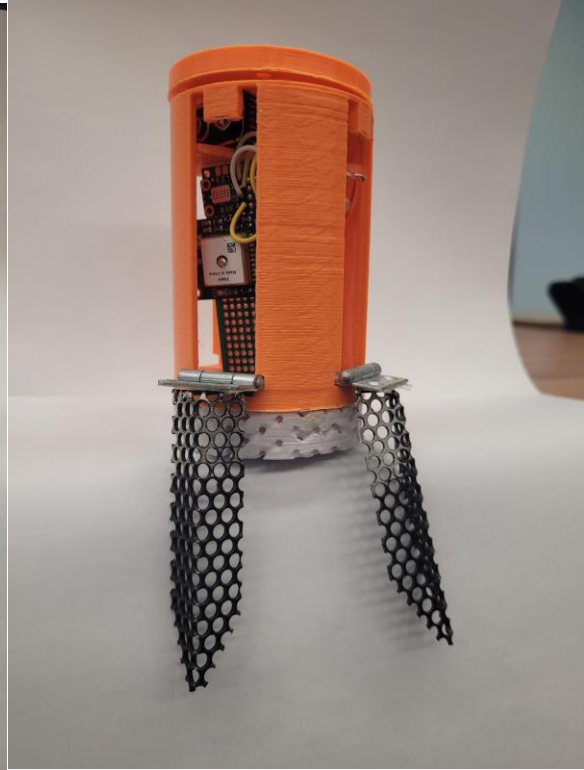
T.I: Participating in the CanSat competition allowed me to refine my technical abilities, apply previous knowledge of python to new scenarios and learn about processes I was previously unfamiliar with; including the implementation of python packages to deal with Arduino data handling. Additionally, I learned valuable lessons about keeping organised as much of the work done had to be achieved before a deadline and required continuous re-evaluation. Furthermore, being able to communicate clearly was paramount to the success of our project particularly concerning exchanges with the Electronics Team as their work was closely linked to that of the Software Team.

APPENDICES

Appendix 1.1: 3D Can Model



Appendix 1.2: Can prototype



Appendix 2: Electronics design

Appendix 2.1: PDR code

```
void setup()
{
  Serial.begin(9600); // make sure serial monitor is set to this rate
  mySerial.begin(9600); // change this to the rate set in if magic or not!
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  bme.begin(0x76);
  //bme.begin(0x38);

  if (!bme.begin(0x76)) {
    Serial.println("BME Sensor init failure");
  }
}

void loop()
{
  if(millis() - timer >= interval) // looping the code to receive live readings of the environment at
  {
    timer = millis();
    //Serial.println("send"); // shows in serial monitor of the transmitter module (TEST)
    //mySerial.println("hello"); // shows in the serial monitor of the receiver module (TEST)
    //mySerial.println(Serial.readString());

    // BME280 Sensor, monitoring temperature, pressure, altitude and humidity (PRIMARY MISSION):
    // Initially code below returned 0 values across the board, but then I ran a I2C port finder progr
    float temp = bme.readTemperature(); // reading the temperature in celsius from the surroundings of
    float pressure = bme.readPressure() / 100.0F; // reading the pressure in hPa from the surroundings
    float humidity = bme.readHumidity(); // reading the humidity in celsius from the surroundings of t
    float altitude = bme.readAltitude(SEALEVELPRESSURE_HPA); // reading the altitude in m from the sur

    //printing live values for the primary mission to the serial port of the receiver module to monito
    mySerial.println("Temperature: " + String(temp) + " °C");
    mySerial.println("Pressure: " + String(pressure) + " hPa");
    mySerial.println("Approximate Altitude: " + String(altitude) + " m");
    mySerial.println("Humidity: " + String(humidity) + " %");
  }
}
```

Code explanation (by I.J.J.):

- I assigned port 10 on the Arduino to be the serial input and port 11 to be serial output by assigning the RX, TX pins.
 - Then I set a loop interval for 1000ms so that the code would run every 1000ms and the sensors would read and output valued every 1000ms to the serial port on the ground station.
 - I then initialized the I2C BME280 sensor by giving the BME280 module the name 'bme'.
 - I then defined a constant, SEALEVELPRESSURE in hPa as 1013.25, to allow us to calculate the approximate altitude of the CanSat during flight by comparing the measured air pressure to that of sea level.
 - I also started by setting both serial objects, Serial and mySerial (the APC220 Transmitter and receiver modules respectively), to the baud rate 9600 so that the data being transmitted would be received on the same baud rate. This baud rate is what would need to be setup on the serial port monitor and python module, in order to read the correct data.
 - Having used the I2C scanner code earlier and figured out that the BME280 sensor was at the location 0x76, I initialised the bme280 sensor at 0x76 with 'bme.begin(0x76)'.
 - In order to error handle in the event that the bme sensor failed to initialise, I used an if statement to output "BME Sensor init failure" to the serial port of the Arduino.
 - Within the 1000ms loop, I assigned the sensor value for temperature to temp, the pressure reading to pressure, humidity reading to humidity and approximate altitude to altitude
 - I then formatted the P.A.T.H. values into doubles with 2 significant figures and output them to the radio receiver mySerial port as a small string, or byte array to store the results.
 - This byte array was then printed to the mySerial serial port to be used in the software.
- [Attached is my GitHub cansat Arduino repository](#), chronicling issues and updates throughout the project.

¹ Jabbar, Arbi Abdul. 2019. How to Scan I2C Address in Arduino. Arduino Project Hub. [Online] September 10, 2019. [Cited: November 3, 2022.] <https://create.arduino.cc/projecthub/abdularbi17/how-to-scan-i2c-address-in-arduino-eaadda>

Appendix 2.2: Electronics since PDR

Figure 1: Testing the MICS-2714 sensor and MICS6814 sensor individually

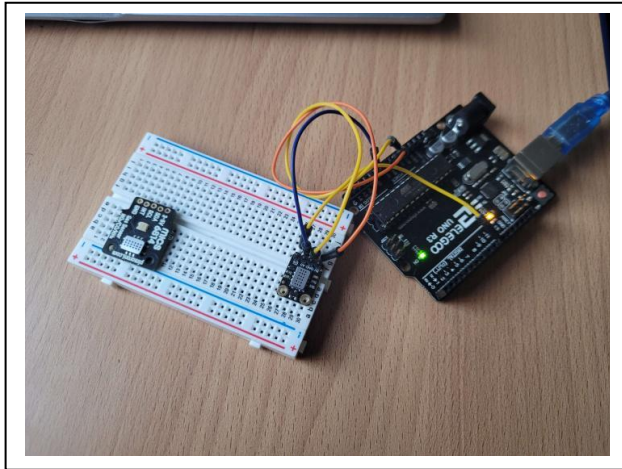


Figure 2: Running I2C scanner to find the I2C address of the SCD41 sensor



Figure 3: Running repeat measurement code on the SCD41 sensor and printing values to the serial monitor

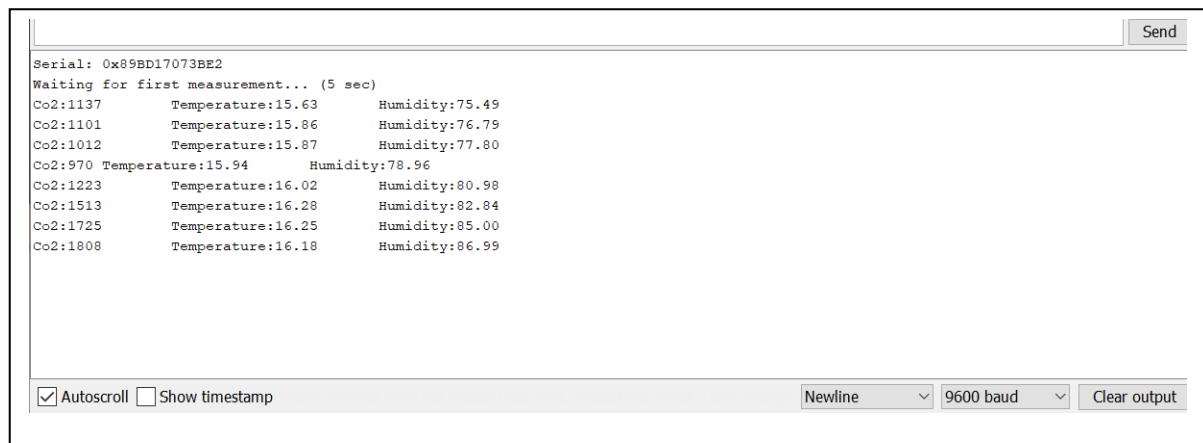
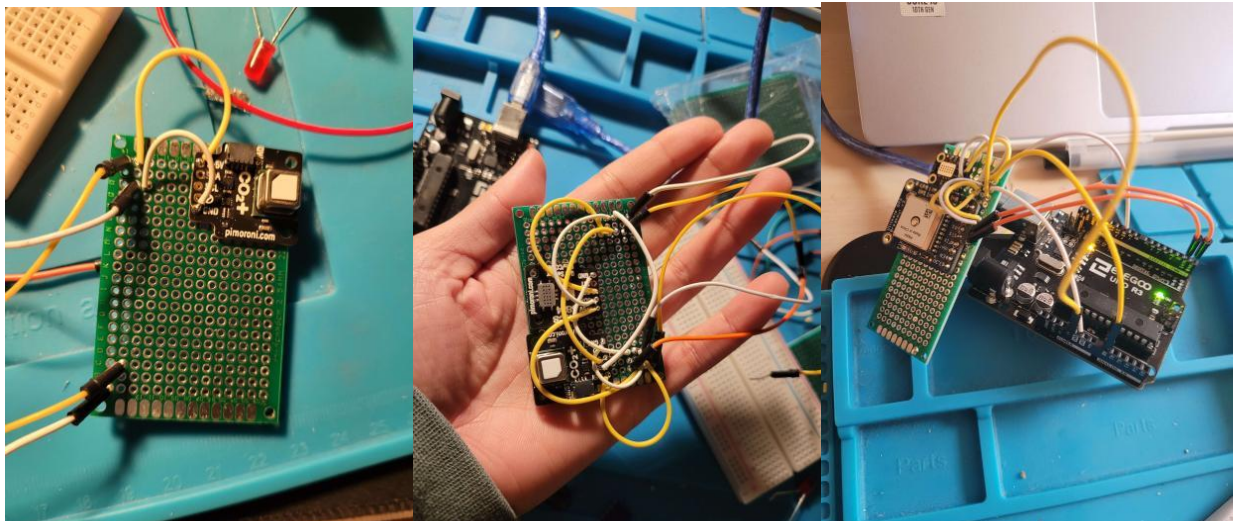


Figure 4: Perfboard in the middle of being soldered, containing CO2 and MICS6814 sensors, followed by perfboard B being tested:



Schematic component credits:

- Adafruit LTR390 sensor schematic: <https://github.com/adafruit/Adafruit-LTR390-PCB>
- BME280 breakout sensor schematic (Adafruit was the only one posted online, but has the same design as the pimoroni version): <https://github.com/adafruit/Adafruit-BME280-Breakout-PCB>
- DFROBOT breakout sensor schematic: <https://github.com/acrobotic/Ai-Breakout-MICS2714/tree/master/pcb>

Appendix 3: Software

Appendix 3.1 Habitability Check code

```
mean_temp = np.mean(tempvalues)
mean_altitude = abs(np.mean(altvalues))
mean_pressure = abs(np.mean(pressvalues))

percentage_humidity = abs(np.mean(humvalues))/2 #mean humidity divided by average
atmospheric_pressure = 101000 #in pascals

def habitability_check():
    if percentage_humidity > 50 and mean_temp > 35:
        survival = False
    elif percentage_humidity < 5 and mean_temp < 0:
        survival = False
    elif mean_altitude > 7925:
        survival = False
    elif mean_pressure > 101000*(57/100): #pressure causing planet to not be habitable
        survival = False
    else:
        survival = True

    if survival == True:
        print("This planet is habitable")
    else:
        print("Human life would not be able to survive indefinitely on this planet.")
```

Appendix 3.2 On-board software

```
//APC 220 code
SoftwareSerial mySerial(10, 11); // assigning RX, TX pins
unsigned long timer;
unsigned long interval = 1000; // setting loop interval for 1000ms

// I^2C BME280 Sensor initialisation
Adafruit_BME280 bme; // giving the BME280 module the name 'bme'
//SensirionI2CSd4x scd4x;
//MiCS6814 mics6814;
Adafruit_LTR390 ltr390;
#define SEALEVELPRESSURE_HPA (1013.25) // defining the constant of sea level pressure in hPa, to allow us to approximate the altitude of the satellite
//SCD4x scd4l;

//mics2714 setup
//#define ADC_PIN A0
//#define POWER_PIN 6
//DFRobot_MICS_ADC mics (/*adcPin*/ADC_PIN, /*powerPin*/POWER_PIN);

void setup()
{
  bme.begin(0x76);
  scd4x.begin(Wire);
  mics6814.begin(0x19);

  ltr390.begin(0x53);
  //Wire.begin();

  Serial.begin(9600); // make sure serial monitor is set to this rate
  mySerial.begin(9600); // change this to the rate set in rf magic or not!
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);

  if (!bme.begin(0x76)) {
    Serial.println("BME Sensor initialisation failure");
  }

  if (!scd4x.begin(Wire)) {
    Serial.println("SCD4l Sensor initialisation failure");
  }

  /*if (!mics6814.begin(0x19)) {
    Serial.println("MICS6814 Sensor initialisation failure");
  }*/

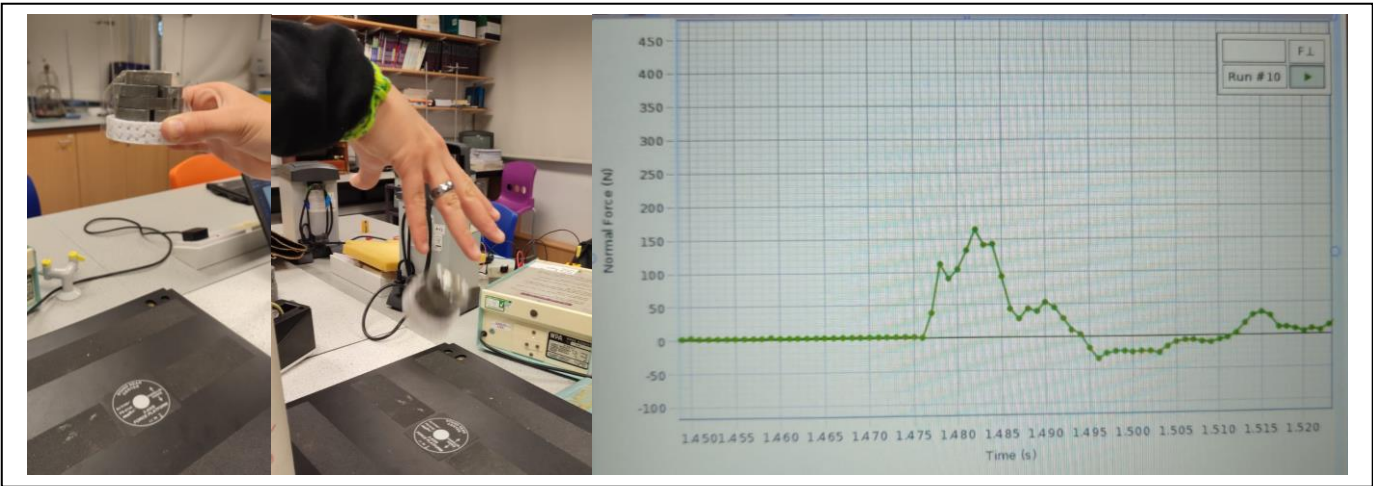
  if (!ltr390.begin(0x53)) {
    Serial.println("LTR390 Sensor initialisation failure");
  }
  //ltr390.setThresholds(100, 1000);
}
```

Landing force test:



Experiment 1:

Impact zone	Force /N				Time /s			
	1	2	3	Mean	1	2	3	Mean
Standard grid	294	268	319	294	0.008	0.009	0.007	0.008
Gyroid lattice	139	230	156	175	0.020	0.012	0.018	0.017
Diamond holes	495	379	371	415	0.005	0.007	0.007	0.006



Experiment 2:

Plasticine shape	Force /N	Time /s
Grid	148	0.015
Circle	234	0.013
Ring	141	0.018

