UK CanSat Competition
Team Athena
Sutton Grammar School
Critical Design Review
Date: 19/02/24

Part 1: Introduction

1.1 The team

We are Project Athena, a 6-man team participating in the CanSat Competition from Sutton Grammar School.

Our roles

Team Lead/Coordinator: Responsible for scheduling meetings, arranging task lists and pushing deadlines. Also responsible for chairing meetings and communicating with different aspects of the team.
Mission/Technical Researcher: Responsible for researching feasibility of mission and equipment needed. Once the plan is in motion, responsible for researching measurements required.
Media Manager: Responsible for directing the outreach programme, including all the project's social media accounts.
Software Programmer: Responsible for writing algorithms in code and handling how the data gets sent/received.
Model Designer: Responsible for creating CAD models and 3D printing the CanSat/other parts.
Electronics Engineer: Responsible for picking components and creating/soldering the circuits. Also helps with the code that runs on the CanSat itself.

1.2 Mission objectives

Our overall mission objective is to simulate a mission to another planet and record data which will be sent back to ground and then stored/analysed.

1.2.1 Primary mission

Our end goal of the primary mission is to take temperature and pressure measurements and send these to ground every second. While the CanSat descends, we will also take VOC (Volatile Organic Compounds) and humidity measurements. These will all be sent down to ground stations and will act as extra data for comparisons/analysis.

1.2.2 Secondary mission: Testing suitability of planet for renewable power production

Our goal of the secondary mission is to test if a planet will be suitable for renewable power production and if so, test which are possible and feasible compared to Earth. The four sources we will be testing for are solar, wind, hydroelectric and geothermal. We selected this mission because right now, humanity is on the precipice of revolutionising how energy is produced so it is vital to figure out if it is possible on other planets too.

The example which sparked this idea was when we read an article about Mars being a viable source of geothermal energy and then discovering other planets had more wind or solar energy than Earth.

Before this mission, we had a previous idea of creating a machine-learning algorithm which detects space junk and the image then gets sent down to ground where it passes through a deconvolution algorithm to get rid of motion blur. However, while researching for parts, we discovered that the four cameras we needed to get a 360 view to detect the space junk were too much for a microprocessor to handle so we had to drop the idea. Luckily, we discovered this very early on as we tried to get a prototype as soon as possible which meant not much time/resources were wasted.

1.2.3 The process of the mission:

1. The shape, weight distribution and legs (specifics detailed later) of the Cansat will make it land upright.
2. CanSat will begin the process of taking measurements and conversion
3. Readings from our sensors will be sent back to ground
4. An script written by us will return a report including:
    a. Reading - Time Graphs
    b. Reading - Reading Graphs

c. Hardcoded values to compare data against

1.2.2 Measurements, rationale and methodology

Our sensors will be sending readings back to ground every second during the descent and every 30 seconds upon landing. These intervals can be controlled from the ground station. The delay in taking measurements is to save battery power and extend longevity.

Measurements and their uses:

- Light (Light dependent resistor):
  - The light intensity will be sent back to ground and will help determine if solar power is viable/effective.
  - It will also show the day/night cycle which will again help determine if solar power is viable/effective.
- Humidity (DHT20):
  - Will reveal if there are any bodies of water nearby and possibly their size.
  - High humidity suggests there is water available which is required for hydroelectric energy.
- Air Pressure (BME680):
  - This is required in the primary mission
  - We can analyse the effects of air-pressure on temperature, humidity and look for correlations.
  - Calculate altitude (formulae detailed later)
- Wind (Custom-made anemometer):
  - Calculate wind velocity which shows if wind power is viable as wind turbines need a velocity between a certain range.
- Temperature (DHT20 & BME680):
  - Record heat patterns which will help scientists make informed decisions on if building these energy sources is feasible and if so, what maintenance would be required for protection.
  - Required for Primary Mission
- GPS (Adafruit PA1616D GPS):
  - This will be used to keep track of the CanSat and can be used for testing purposes as we can track the flight path.
  - It will also help us find it afterwards.
- 9-axis Inertial Motion Unit (ICM-20948):
  - Gives all around useful information such as if the CanSat is descending correctly and hasn't reached an uncontrollable spin.
  - Can be used to track which way the CanSat is oriented.
  - Can be used to track the acceleration forces on the CanSat during different stages of the launch.
  - Can be used to calculate the force the CanSat hits the ground to see if the parachute works correctly .
  - Will be used to determine is geothermal energy is possible as we can detect earthquakes
- Volatile Organic Compounds (BME680):
  - This will be taken during the Primary Missions and will allow us to analyse any correlation of the readings with temperature, altitude or humidity.

A plan of our data analysis can be found in Appendix J.

Part 2: Project Planning

2.1 Project timescale

2.1.1 Meeting schedules

We had a general meeting every Monday at lunch in school, and multiple flexible drop-in or online sessions through the week. In those meetings, we discussed work schedules, CanSat design tweaks and proposals, and sometimes even built and designed parts.

2.1.2 Task arrangement

We grouped our tasks into Work Cycles, each lasting a week. Below is an outline of our work schedules, agreed at the beginning of every Work Cycle.
Appendix I contains a high-level Gantt Chart displaying our task schedules between the PDR and the CDR.

2.1.3 GitHub

To make our coding process run as smoothly as possible, we used Git for source control. Our code is broken down into two repositories, one for the code on the Pico and the other for the ground station code. This also allowed for easy tracking of progress and through the use of GitHub Projects and allowed us to plan ahead easily. Our software engineers were regularly committing code with informative comments and nearly all of our code has comments to make it easy to understand. By using Git, it was easy for multiple people to work on the same project.

Our GitHub: https://github.com/CanSat-Athena

2.2 Team and external support

2.2.1 Team skills

We are a highly dedicated team of 6, and one of two teams representing Sutton Grammar. The majority of us have previous experience from working on the Big Bang competition last year and so we have a strong team culture and bond between members. We are especially able to adapt to sudden changes of plans. For instance, at the beginning of our project we had a really good secondary mission idea that would mean we had to spend about a week learning necessary skills for the mission. However, about four days in, when we were running feasibility studies we found that no equipment advanced enough was compatible with the Raspberry Pi Pico we were using. The Team Lead quickly coordinated multiple emergency meetings and we successfully managed to repurpose our secondary mission within 24 hours. More recently, the team had a tight timescale to work with producing the antenna and parachute systems and testing due to all the parts arriving late, and a massive shopping list. The team worked round it by quickly changing our testing plans to accommodate the late arrival of parts, including a lot of partial testing. Through these incidents, the team has repeatedly displayed its adaptiveness in urgent scenarios and quick thinking.

2.2.2 Technical skills

The majority of the team is maths-based, with a long history of participating in STEM-related competitions, including the 2022-23 Big Bang Competition, in which some of the team had to learn an entirely new programming language for our project. This would massively help achieve our objectives of the Secondary Mission, which requires us to process received data and determine whether our simulated environment is hospitable or not. As mentioned before, multiple team members can program in C/C++, Java and Python and are always keen on experimenting with different mediums of interpreting and constructing algorithms, functions and systems. Our lead electronics engineer is highly experienced with the Pico and its workings, having done many projects previously.

2.2.3 Limitations

Our team is very strong on a lot of aspects, but one thing that the team is short on is someone who does graphic design. This is mainly regarding the outreach poster and logo designs, so it is not as much of a problem. Also, to learn the concepts needed to engineer our CanSat and program our missions, the team members need time to learn and master the concepts, as many are A-Level content and all our members are still doing GCSEs. However, the team is enthusiastic and very flexible, so this should not prove too much of a limitation.

2.2.4 External support

Our team has an expansive external network, including plenty of the team members' family friends being engineers. One of them, a mechanical engineer, helped guide us to the idea of our exterior design of the CanSat and introduced us to the roly-poly mechanics behind the design, though we ended up abandoning the idea after testing.. Another, a meteorologist, helped us come up with the methodology for our wind and solar measurements, which we are taking for our Secondary Mission. At the same time, our peers have significantly helped us in the project as well. As one of our team's weaknesses is poster and logo designing, we outsourced to one of our classmates to help design our logo (which can be seen on the cover). On top of that, we worked with another team from our school who is also part of the competition and we traded services. One of their members was experienced at drawing so she drew our sketches for us and in return, one of our members knew how to code in LaTEx so he coded their maths equations into their document. The team also has advisors in upper years who have experience in mechanical design, and will help us peer-review our reports and designs in order to help us further improve and add to our output.

2.3 Risk assessment

2.3.1 Physical

Short circuiting: Exposed wires can contact other parts of the circuit, causing high current draw, breaking the affected components and/or causing unintended side-effects. To mitigate these risks, we will be securely mounting each component, using JST connectors and using heat shrink tubes/hot glue to isolate exposed electrical connections.
Soldering/3D printing fumes: Soldering and 3D printing can release toxic fumes that are harmful to the lungs. To mitigate this risk, we will be soldering in a well-ventilated area, using a fume extractor and using lead-free solder. When 3D printing, we will be printing in a ventilated area and avoid using filaments that produce an excessive amount of fumes.
Soldering/3D printing burns: Solder and 3D printing filament melt at 200°C, which can cause burns if touched. We will avoid contact with the 3D printer nozzle while the machine is in operation and will take care when soldering by keeping a perimeter around the solderer where no-one else is allowed.

2.3.2 Technical

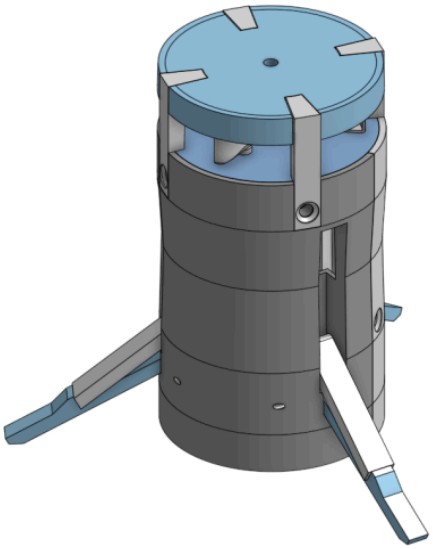
Time constraints: We decided to take part in this competition in mid September and we hadn't anticipated the amount of school work we would be faced with. This is because our team has just started their GCSEs and most are taking 11, with some taking 12. This has led to an unforeseen rise in work and this rise will only increase as we approach our mocks. To mitigate this risk we have done two things. Firstly we chose our Secondary Mission as something which is achievable without the need to dedicate an extraordinary amount of time. Secondly we have also planned our tasks with a generous amount of buffer time which will greatly help when our work suddenly increases.
Size/weight constraints: Due to the nature of our secondary missions, size and weight is extremely important and there is a chance that some electronics don't fit in which has the potential to render parts of the missions impossible. To combat this, we have selected electronics which will be the first to be cut if need be as their removal affects the mission the least. There is also a risk that our 1200mAh battery does not fit. We discovered this through modelling in CAD and we found out that it is a very close fit. For this reason we have identified another battery which we can use as it is smaller due to it being only 750mAh but this will be a last case resort.
Anemometer not working: A quarter of our secondary mission relies on detecting wind speed but there is a significant chance that our anemometer will either not detect velocity accurately due to its size or break on impact due to its intricate design. This is probably one of the greatest potential risks so we have started making the anemometer in

mid-December and got a working prototype done by the end of December. Then we carried out extensive tests (see section 3.6.1).
CanSat is prone to breaking: There is a risk that the 3D printed parts may be damaged during the launch. To mitigate this risk, we are using Polycarbonate printed at a high infill level and 3+ perimeters. We will also design the parts in a way that allows them to be printed in an orientation where the stress applied is perpendicular to the layers of the 3D print, lowering the chance of layer separation.
Defective Components: There is always a risk that a part we order comes with an issue, therefore we will have spare sensors on the launch day and will design the electronics in a way that lets them be replaced quickly by using JST-PH connectors (see section 3.3.3).
Antenna Signal disruptions/transmission failures: This is perhaps one of the most concerning risks of all, as transmission failures would hugely hamper the success of the missions. As such, we will do thorough testing in order to make sure the antennas can successfully transmit signals to and from the CanSat and ground station (See section 3.6.5 for details).

Part 3: CanSat Design

3.1 Mechanical design

3.1.1 General overview



3D Model of our CanSat, using Onshape

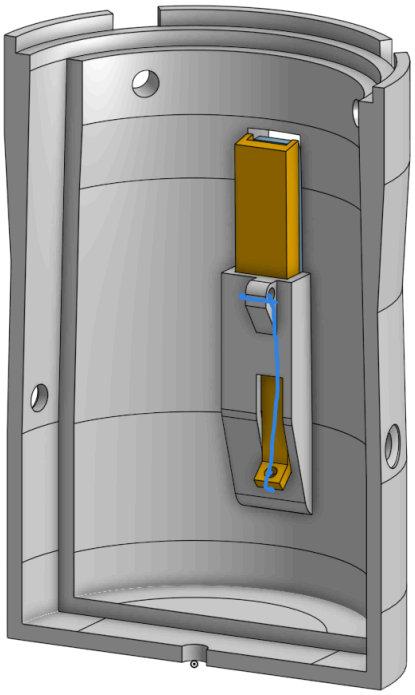
Real-life Model of our CanSat

As we have switched to the leg uprighting system, we decided to abandon the self-balancing curved bottom mechanism for two reasons. Firstly, it would reduce the amount of space available inside and was unnecessary as we are using the leg mechanism. Secondly, in our testing, when used in combination with the leg mechanism it had the opposite effect, making the CanSat more likely to be tipped over.

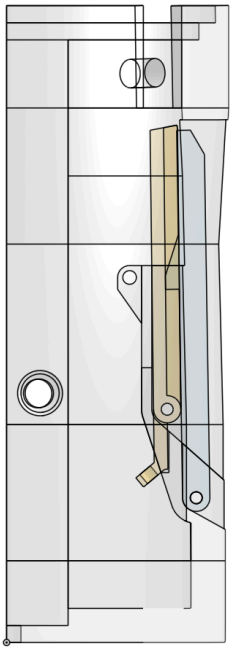
3.1.2 Leg uprighting system

One of the most significant changes we made since writing the PDR is the re-introduction of a (modified version of the) leg uprighting system for our CanSat. We decided to switch away from the previous uprighting system that relied on a hemisphere and weight distribution, turning the hemisphere into a flat base with round edges. One of the main reasons for doing so was because it greatly reduced the volume available inside the CanSat. In the PDR, we determined a leg

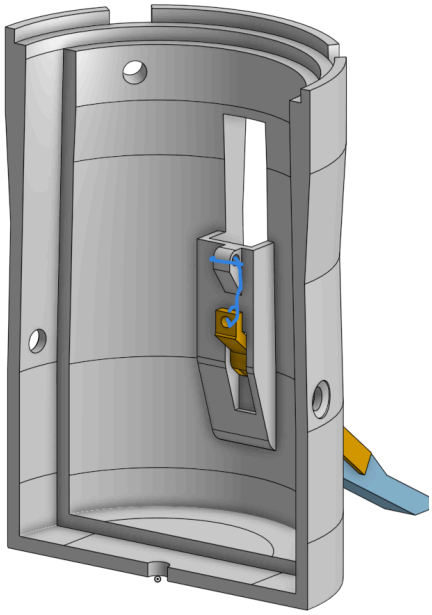
system with a motor as a risk that would increase the chance of failure. The team still agrees with this statement, and therefore the new leg design will rely on elastics, and not a motor to extend. In the team's opinion, this new system would be more feasible to execute and has a higher chance of success than using the previous uprighting system or a leg system with a motor. As of right now there are three legs but this may be increased to four if there are balancing issues.



3D view of the folded leg system



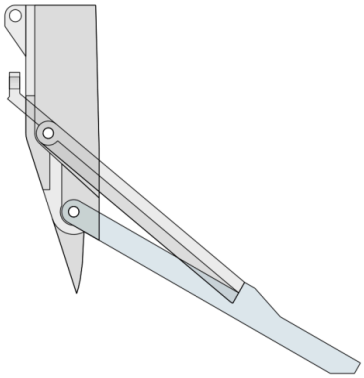
2D view of the folded leg system

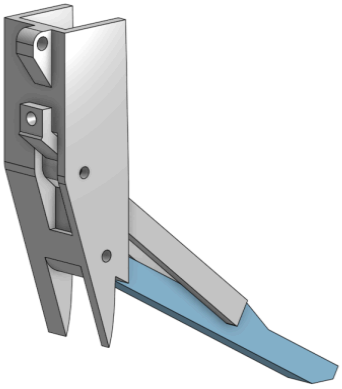


3D view of the unfolded leg system
The blue lines drawn on in both 3D images represent the elastic material

On the above left image, you can see that the elastic band would be connected from a notch on the top of the fixed component (in grey) to a notch from the extension (in yellow on left image). As the CanSat leaves the rocket, the legs would extend as the yellow notch would be pulled up, pushing out, dropping and extending the folded leg (in yellow, above both notches) down into the position seen above right.

The notch being pulled up is connected to the top part of the leg (in yellow, like the previous image). Pulling the notch up nudges the top part of the leg, which is connected to the bottom part (blue) at its tip. The top part of the leg tries to pop out, pushing the bottom part out as well, and the two parts slide into place.



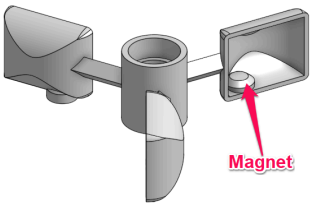


2D view of the extended leg system

3D view of the extended leg system

3.1.3 Anemometer

A key part of our secondary mission is that we will be measuring the wind speed. To do that, we will be using an anemometer. It is 12mm tall and has square cups to make the most of the available wind power. We had to 3D print ~15 different anemometers to get the cup width and depth just right to get the maximum possible RPM and measure using a custom-made mini wind tunnel (see section 3.6.1 and Appendix C). To be able to measure the rotation speed of the anemometer we will have small cylindrical magnets on the bottom of each cup and a hall effect sensor to sense the magnets (see section 3.2.1: Hall effect sensor).



3.1.4 Materials Choice

We chose Polycarbonate filament to 3D print our CanSat because of its high strength compared to other filaments. The polycarbonate filament has high impact resistance which will be helpful in maintaining the structural integrity of the CanSat when it lands and help it survive the high acceleration forces during launch. It is also relatively cheap compared to other filaments at £22.00 per kilogram. Furthermore, it has strong layer bonding, which means that it has high tensile strength perpendicular to the layer lines as well as parallel. On the other hand, there are some disadvantages to polycarbonate filament as well. It is quite hard to 3D print for starters and it has a tendency to ooze while printing. The oozing could result in a weaker structure; we may even have to re-print the part again. It is also prone to warping which could decrease the accuracy of the print. Finally, it needs a very high printing temperature (260+°C) and the 3D printer we are using can only go up to 255°C, so instead we may use Tough PLA or PLA+, as it is the second best filament for our use case and does not require such a high printing temperature. We have researched further into other filaments as well and we have the results of our research in this PDF:
https://cansatathena.com/wp-content/uploads/2024/02/Filaments.pdf.

3.1.5 Threaded Rod

We are using a threaded rod which runs directly through the centre of the CanSat as a ‘spine’ to provide strength. As metal has much higher tensile strength than plastic, having the parachute connected to this instead means that the metal will take most of the force instead of the plastic. This ensures that the CanSat doesn’t break into pieces when the parachute opens.

3.1.6 Special Printing Methods

In addition to previous choices, we have adopted a special printing method that would allow us to embed nuts into the 3D print. During printing, the 3D printing process will pause halfway in order for a nut to be placed in. Then, the 3D print will resume, and will print over the nut. This would secure our CanSat more effectively by locking small parts into place, and also increases our CanSat strength as a whole.

3.2 Electrical design

3.2.1 Components

A list of the components and prices is available in Appendix H

Pico LiPo (16MB)

We are using the Pimoroni Pico LiPo as our microcontroller. This is for a few reasons. It is small and lightweight which is crucial in a space and weight limited mission like this. The size also gives more room to play around with when deciding where internals go for the correct centre of gravity. For its size, it also has a lot of pins which is necessary as we will be using a lot of sensors. We are using this instead of the regular Pico as it has a 8x greater flash memory which will allow us to store plenty more readings. It also allows for the use of a LiPo battery without a separate battery charging circuit, which would take up valuable space and we are already space-constrained due to the anemometer.

BME680

We had a choice between this sensor and BMP280 as they both measure pressure. Initially we were going to choose the BMP280 as it was already included in the pack however the BME60 also measures VOCs which opens a lot of doors for analysis in the Primary Mission. This swap also doesn't lose any accuracy, both measure pressure with ±1 hPa accuracy and the dimensions are almost identical so it doesn't affect our arrangement.

Adafruit PA1616D GPS

There were many options for the GPS sensor but we chose the PA1616D for multiple reasons. Firstly it has a coin cell battery slot which eliminates "cold starts" which saves us roughly 30 seconds every time we start the GPS. This also has a very high max velocity, 515m/s, and has 3V logic level making it compatible with the Pico (3.3V). Its accuracy is quite high with it being accurate to ± 5m. This is one of the most energy demanding sensors we use but that is common with all GPS sensors and this has one of the lowest current draws with it being only ~29mA during navigation.

DHT20

Even though we already have a temperature and humidity sensor (the BME680 also senses humidity and temperature in addition to pressure and VOCs) this is more accurate so given that we have the budget we decided the extra space it takes is worth it. However, if we find out the internals don't fit, the DHT-20 will be the first electrical device we cut.

ICM-20948

The main reason we chose this as our Inertial Motion Unit (IMU) was that it is very cost effective, especially for a 9-axis IMU at only £14.70. It is also very small, accurate and can measure up to 16g which is plenty for the mission.

Light dependent resistor

It was between this or a photodiode and we chose the LDR because it was cheaper, lighter and smaller. However this comes at the slight cost of speed (photodiodes take mere nanoseconds to adapt to change in light intensity) which isn't that significant as we don't need to take readings at that speed.

Hall effect sensor

As explained in section 3.1.2, we will have magnets on the bottom of the anemometer. Therefore, we need a sensor to sense the changes in the magnetic field. For this, we have two options: reed switches and hall effect sensors. However, hall effect sensors are the clear winner here for three reasons. Firstly, reed switches are made of glass, which makes them extremely fragile and prone to breaking, especially since it will be shot up 300m in the air and will experience upwards of 20G. Secondly, they are mechanical and mechanical sensors usually have a much lower lifespan than solid state ones as the metal inside can wear out over time. Thirdly, reed switches are slower to switch which can be an issue if the wind intensity is high. To measure the number of times the magnets pass over the sensor, we are using interrupts that are triggered on the rising edge of the signal from the sensor. Once the interrupt is triggered, an integer variable is incremented and every second, this variable is reset to zero and the old value is recorded as the 'third of a revolutions' (as the anemometer has 3 cups) per second.

3.2.2 Schematics

The schematics are in Appendix A and a high resolution version is available at https://cansatathena.com/wp-content/uploads/2024/02/main-schematics.png

3.3.3 Modularity

If a component breaks on the launch day, we will have at least one replacement ready. However, replacing components still requires desoldering/resoldering wires correctly, testing that everything is connected correctly and there aren't any shorts - it takes a lot of time. This is why we decided to use modular connectors - we will solder JST-PH connectors to every component which will make swapping components effortless and easy. This means we also will not have to check the connections and check for shorts, making the process extremely fast.

3.3 Software design

Note: All of our code is open source and is available here: https://github.com/CanSat-Athena/CanSat and here: https://github.com/CanSat-Athena/data-processing. The diagrams used (in .drawio format) are available at https://github.com/CanSat-Athena/Diagrams.

3.3.1 On the CanSat

Language choice

For our language, we had two options: C++ or MicroPython. In the end we decided to choose C++ for a few reasons. Firstly it runs at a significantly higher speed (up to 46x faster than MicroPython and 69x faster than CircuitPython - see Appendix D) which is useful when taking real time measurements as our CanSat can't afford long delays due to its speed. Secondly, C++ is a compiled language compared to MicroPython being interpreted. This means that nearly all errors in C++ will be caught at runtime; this is vital in a mission where we won't have access to the device for a long time and can't afford a breakdown mid-descent. Due to being constrained to a microcontroller, memory is an issue with us only having access to 256 KB of RAM and 16 MB of flash memory (we are using the 16MB variant of the Pico LiPo). This is where again C++ comes ahead of MicroPython as it is memory efficient since it gives control over the memory to the user with the use of things like raw pointers. Our electronics engineer has written a comprehensive article about the points above on our blog here:
https://cansatathena.com/2024/01/25/why-we-are-using-c-over-micropython-for-our-cansat.
Finally, due to the need of having to schedule different parts of the code at precise intervals, a tool which helps with that will be greatly beneficial. This is where C++ opens the door for us to use FreeRTOS (Real-Time Operating System). There aren't very many practical real-time operating systems in the MicroPython ecosystem, therefore we must use C++ to be able to take advantage of such tools. This allows us to multitask easily even with just the 2 cores of the RP2040 chip and have a different, separate task for each component of the code, all running at the same time with FreeRTOS's powerful task scheduling system. Real time operating systems are used in NASA missions and practically all space missions which speaks to their usefulness.

The fact we are using C++, an OOP language, also gives us many advantages. Firstly, given our mission requires accurate data readings, the data corruption must be kept to a minimum. This is where one of the fundamentals of an OOP comes to play, encapsulation. By encapsulating, it makes it less prone to bugs. However the biggest advantage is that it makes the code organised and modular. Through the use of classes (and sometimes structs) the code is broken down into independent sections allowing for easy troubleshooting. It also makes the code flexible and allows us to add code very easily and not have to worry too much about breaking everything. By using inheritance, a lot of time is saved as code can be reused. An example of this is when our software engineer wrote just one function for reading data on the I2C bus and a parameter with the sensor's details instead of having a separate function for each sensor. Even though Python has classes, C++ has a better implementation of them.

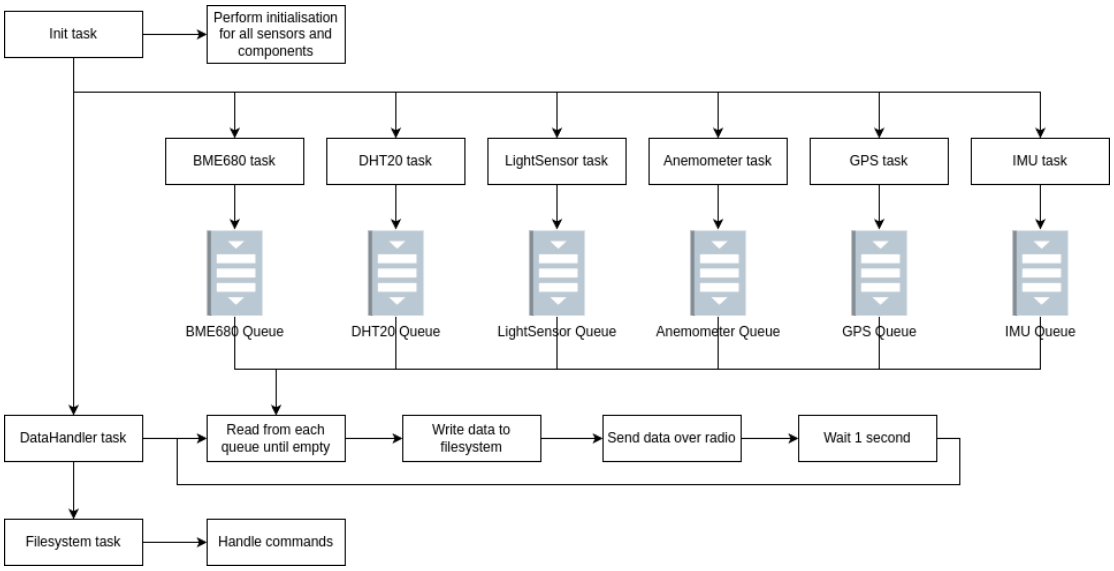
FreeRTOS design

As we are using FreeRTOS, we have designed every component of the code as a separate task, using queues to interface between them. Firstly, when the CanSat is turned on, the initialisation task runs. Inside this task, the `init()` method of each component's class is called and the tasks of the components are started.

As each sensor task performs a similar task - read data, wait for an amount of time, repeat - we have made a single task function to handle all the sensors. Each sensor still has its own task, however as the FreeRTOS method `xTaskCreate` has a parameter for passing values to the task (`void* pvParameters`), when we are creating the task we can use the same

function for all sensors and instead pass a pointer to a struct (another reason to use C++) that contains data about the individual sensor. The sensor data struct has data such as the instance of the Sensor class, the FreeRTOS queue to send the sensor data to and the measurement delay period. In the sensor read task, there is an infinite loop; inside the loop the read function of the sensor class (as said, passed as an argument) is called, the returned data is added to the FreeRTOS queue (again, passed as an argument) and the task sleeps (`vTaskDelay`) for the sensor's delay period. Implementing it as such instead of having a separate task function for each sensor means that it requires relatively little changes to the code to add or remove a sensor, making the code modular. Another advantage of this is that if we want to make a change to the function because of, for example, a bug, we only need to make changes in a single place instead of six.

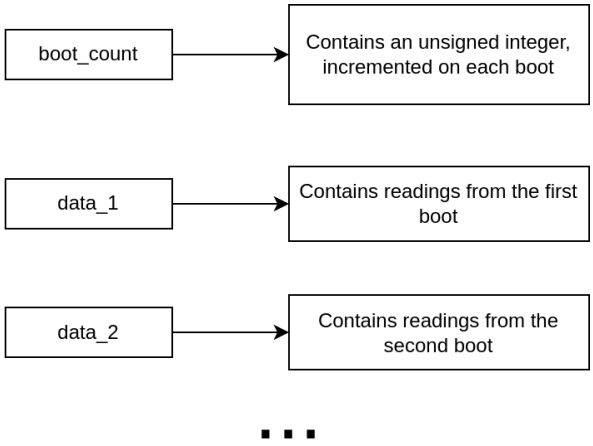
Each sensor's task adds its data to its queue and we need to send and store that data. Therefore, we need a task to handle that data, which is exactly what the data handler task is for. It is responsible for reading the data from the queues, storing it in the filesystem and sending it over radio to the ground station.



Filesystem

We decided to ensure that, in the rare scenario that the radio signal was lost, the data wouldn't be lost too. Therefore, we have decided to store the data on the CanSat. We considered using an SD card, however as that added unnecessary cost and complexity we opted to use the built-in storage of the Pico. As we are using the 16MB variant of the Pico LiPo, we came to the conclusion that 16MB was more than enough - it would allow us to store readings every second for at least 6 hours and we can free storage remotely. At first, we considered storing the data on the raw flash, however as that would be difficult to manage, provide no flash wear levelling and provide little to no power loss resilience we decided to instead use a filesystem. We did have a few options such as FatFS and SPIFFS but in the end we decided on LittleFS due to its excellent dynamic wear levelling, power loss resilience and small RAM and ROM usage, making it excellent for embedded applications.

In the filesystem, we will have a file called `boot_count` that stores the number of times the Pico has been booted. We do this by, inside the initialisation function of the filesystem, opening (or creating if not present) the `boot_count` file, incrementing the value, and writing the value back to the file. The boot count is then used to create a data file, `data_[boot count]`. This ensures that if the Pico unexpectedly resets due to, for example, a hard fault or the battery wires getting disconnected momentarily, the old data will never be overwritten. As a result, this, combined with the power-loss resilience of LittleFS, acts as a fail-safe to ensure that it is near impossible to lose data.



▪ ▪ ▪

Fail-safes

- We are using the Pico's built-in watchdog feature (as described in section 4.7 of the RP2040 Datasheet here: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf). A watchdog is a countdown timer that resets the RP2040 when the value reaches zero. During normal execution, the watchdog is updated periodically (by calling `watchdog_update()`), however in the case of a rare event such as a FreeRTOS kernel crash the pico will reset. This is vital to ensure the CanSat will always remain operational, even if there is a crash.
- No data is overwritten on a reset because a new data file is created on each boot, as explained above.
- Vital tasks have a higher FreeRTOS priority than less important ones, for example the data handler task (handles ground communications and storage) has a priority of 3, whereas sensor read tasks have a priority of 2. This ensures that if one of the sensor read tasks is stuck in an infinite loop, the more important data handler task will not be stuck in a blocked state.

3.3.2 On the ground

Receiving data

To receive data, we are using another Raspberry Pi Pico and an RFM9X radio to receive data from the CanSat. The Pico will print out the data via the USB serial, which is read and parsed by the Python program.

Parsing

The data from the CanSat is sent as a line of CSV, transmitted every second. We debated on using JSON format instead but went with CSV due to the fact that it uses less characters, therefore using less bandwidth at the cost of readability. This CSV format is then read by a Python program that iterates through each line of CSV and parses it every second. The code also keeps track of how many linesit has parsed to avoid duplicate data entries. The program categorises which table the data goes into (will be explained in a moment) then reads the value of the reading. Using the `UPDATE` SQL command, the program then adds the data into the database table and then goes onto the next reading and repeats the process. To parse the data, it iterates through each value and references a format to see what data type it is, for example if it is the third value in a line that value will be given the reading type of accelerometer data. The reading type, reading value and along with the time (which is the first value in the line) gets added to the table. To see what table the data goes in, the program checks for the reading type in a list and then finds the correct table. If the reading type goes in the Average Database, it queries for the previous values from the type and then finds their average. Once the program finds the new line character (`\n`) it starts the parsing process again as a new set of data has been received.

A flowchart of this is available in Appendix B.

Data storage

We are storing our data using SQLite3. We used SQLite3 instead of mySQL as it is very light-weight and seeing as only one user will be accessing the database, there's no need for the complexity of mySQL. In our database there are 5 different tables: Average Table, IMU Table, Regular Table, Regular Adjusted Table and Average Adjusted Table. Each table has three columns: Reading Type, Reading Value and Timestamp. The reading type is what sensor it came from and what it read, e.g bme_pressure. The reading value is a float which just stores the value and the timestamp stores the UNIX time when this was taken. The average table stores the running average of certain readings. The IMU table stores the accelerometer, gyrometer and magnetometer readings and the regular table stores the rest. We are storing the IMU readings separately because one reading has multiple values so it is easier to store it this way. The two adjusted tables store the exponential adjusted data (detailed later in data conversion). For the timestamp, we are using the time since the Pico's boot which is in milliseconds. However this is only the time the CSV is sent and does not account for multiple readings taken per second. To combat this we have to figure out separation time by dividing the difference between the time of the current readings and past reading by the number of readings taken. Now this value is subtracted from the time the CSV was sent for every reading. For example, if during one set of data which was sent 60000 milliseconds after boot there are 5 dht_temp readings and the previous set of the data was sent at 59000 milliseconds, the scripts will subtract the two times to get 1000 milliseconds. Then it'll divide 1000 by 5 to get a separation time of 250 milliseconds. So when it enters the values into the table, the first one will have a timestamp of 60000, then the seconds one will be 59750 then 59500, then 59250 then 59000. Even though this doesn't give the exact time as it assumes equal spacing between readings, it's a good estimate.

Data conversion

When the ground receives data, it will carry out a few conversions/calculations. Firstly it will calculate the altitude of the CanSat, even though the GPS calculates altitude it is not accurate enough for our needs, using the air pressure. We will use the formula below:

$$\text{Alt} = \frac{10^{\frac{\log_{10}(\frac{PF}{PS})}{5.2558797}} - 1}{-6.8755856 * 10^{-6}}$$

Where PF is the pressure measured by the CanSat and PS is pressure at sea level which we will assume to be constant. This formula gives the altitude from sea level so we will have to subtract the field's altitude to get a relative altitude.
The hall effect sensor will return the amount of time the magnet has passed it in a second so to calculate the speed, the script will multiply the diameter of the anemometer circle by that amount.
As mentioned in our data analysis, the wind speed at 85 m is what we want for analysis so the script will apply the logarithmic shear formula to get the useful speed. To reduce the impact of outliers on the data we will be calculating the exponential moving average for nearly all the data as well.

Data presentation

To present our data, we will be using Matplotlib. Matplotlib was chosen for its ease of use, compatibility with Python and its integration with Numpy and Pandas. Using a text based GUI, a user can choose between a reading-time graph or a reading-reading graph. If a reading-time graph is chosen, the user can enter what reading they want (e.g dht_temp) and if they want the average or regular data. The script then reads from the aforementioned database and queries for the entered data type in the entered table then stores each value into a list along with the timestamp in another list. These two lists are then used as the data for the graph which gets displayed. The user also has the choice to make a title and label the axes. If the reading-reading option was chosen, the user enters the two data types they want. The script then queries for one data type and stores its value into a list. It then looks at the timestamp of the reading and queries for that timestamp and the second data type in the database. This value is now stored in another list. Now there are two lists of readings taken at the same time. These lists get used as the data for the graph and again the user can choose the title and data labels.

A flowchart of this is available in Appendix F

3.3.5 Descent stage

During the descent stage, humidity, air pressure and temperature will be taken multiple times a second and will be sent to ground in CSV format. On ground this data will be adjusted and further calculations will be made such as measuring altitude, this is detailed earlier in data conversion.

To keep exact track of how the CanSat is doing, on ground checks will be carried out to see if the CanSat has been ejected, if the parachute has deployed successfully, if the terminal velocity has been reached successfully and if it has landed.

To check if the CanSat has been ejected, we will use the accelerometer readings. The CanSat will experience roughly 15G of force of $147.15ms^{-2}$ in acceleration. A Python script on the ground will be checking the acceleration readings sent by the CanSat and will be comparing the current one to the value 5 readings ago and if there is a decrease of at least 100 ms^{-2} (roughly 10g) a message will show saying “CanSat Ejected”.

To check if the parachute has been deployed the accelerometer reading comparisons will continue and if there's a sudden decrease of 50 ms^{-2} (roughly 5G) a message will show up saying “Parachute Deployed Successfully”. If there is no sudden decrease after 15 seconds, a message will show up saying “Parachute Not Deployed Successfully”. The time limit and exact thresholds are subject to change after further testing.

To check if terminal velocity has been reached, 5 seconds after the parachute has been deployed successfully, acceleration readings will be checked. If they are less than $1ms^{-2}$ then a message saying “TV reached successfully” since at terminal velocity the acceleration is 0 (we allow anything less than 1 m to account for accelerometer inaccuracy).

To check if the CanSat has landed, 55 seconds after it has been ejected the altitude will be checked and if it is less than 1m a message will show up saying “CanSat Landed: Descent Over”, this check will be repeated until it has landed. 55 seconds was chosen since after using the SUVAT equation $s = ((v+u)/2)t$, by solving for it we calculated the CanSat to take 75 seconds to hit the ground so 55 seconds is 20 seconds before just in case it falls faster than expected.

A flowchart of this is available in Appendix G

3.3.6 Post-landing stage

After landing, the CanSat will enter a low power mode to save energy. Instead of taking multiple measurements every second, measurements will be taken once every 30 seconds and the GPS sensor - the component with the highest power consumption - the GPS sensor - will be disabled, the RP2040 and other sensors (where applicable) will be put into deep sleep mode, only waking by a timer interrupt every 30 seconds.

3.4 Landing and recovery system

3.4.1 Parachute design

To ensure the highest chance of our CanSat not taking damage and having a smooth landing, we have opted for our CanSat to descend at a terminal speed of 8.5m/s. Our descent speed has altered from our PDR because we were referencing the 2022-2023 guide instead of the newly updated one which led to a wrong understanding of what the expectations were for descent speed.

Our final parachute design will be a flat-pack hexagon design. We have chosen this design for a few reasons. Firstly, it is simple and relatively easy to make which allows us to focus our attention on more vital parts of the mission. Even though it is easy to make, it has a relatively high drag-coefficient of roughly 0.80. This is useful since our descent speed is quite low, a high drag coefficient allows us to achieve that speed with less material/area, a vital advantage when dealing with space/weight limitations.

After the PDR, we entertained the possibility of doing a parafoil design. We planned to have the parafoil cords attached to two servo motors which would have allowed it to steer on command. However after some brainstorming/discussion,

we decided against parafoil and stuck with our flat-pack design as research showed it would be a big task and seeing as our secondary mission wouldn't benefit much from it, there was no point.

We used the drag and weight formulae to calculate the required area of the parachute:

$$\text{Weight force} = mg \text{ Drag Formula} = \frac{1}{2}pv^2C_dA$$

Where m is the mass of the Cansat, g is the gravitational constant, p is the local air density, v is the velocity and C_d is the drag coefficient.

Assume the mass of the CanSat is 0.35kg, p is 1.225gm^{-3} , the velocity is 8.5ms^{-1} and the drag coefficient is 0.8. The terminal velocity occurs when the weight force is equal to the drag force so we can set the equations equal to each other and solve for A

$$mg = \frac{1}{2}pv^2C_dA \Rightarrow 2mg = pv^2C_dA \Rightarrow A = \frac{2mg}{pv^2C_d}$$

Then plug in the assumed values frome before and get:

$$A = \frac{2(0.35)(9.81)}{(1.225)(8.5)^2(0.8)} = 0.0970m^2$$

Therefore our hexagon has an of area 0.0970m^2

We can use the area of a hexagon formula to calculate the side length:

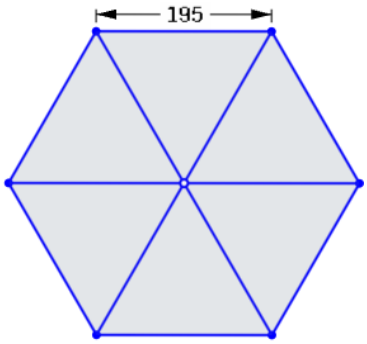
$$A = \frac{3\sqrt{3}}{2}a^2 \Rightarrow a = \sqrt{\frac{2A}{3\sqrt{3}}}$$

Plug in A = 0.0970:

$$a = \sqrt{\frac{2(0.0970)}{3\sqrt{3}}} = 0.193m = 19.3cm \approx 19.5cm$$

So after calculations, our parachute will be a hexagon with side 19.5cm or 195mm. To make the making easier, we split up the parachute into 6 equilateral triangles of side length 19.5cm or 195mm.

The blueprint:



Materials

For our parachute material, we have gone for nylon ripstop fabric. This is for a few reasons. Firstly it's easy to obtain and customisable so we can make it into our own design easily. As it is woven with coarse, strong warp and filling yarns the tears don't spread decreasing the chance of failure. The major reason we chose this material is because it is waterproof, tear resistant and has zero porosity which reduces the chance the actual CanSat gets extremely wet. Finally, it also has a high strength to weight ratio which favourable as in this competition every gram matters.

For our parachute cord material, we chose a paracord made from nylon with a diameter of 4mm. Due to its 11-threaded core it can withstand 3000+ Newtons, which is plenty enough for the CanSat.

3.4.2 Recovery system

To ensure the highest chance our CanSat is recoverable, we have opted for three systems in place. Firstly we will use GPS to know exactly where our CanSat is, this almost guarantees the CanSat doesn't get lost and has the added advantage we can track its flight path which is useful during testing. However, since our GPS is only accurate to roughly 20m, we have made our parachute and the cord a fluorescent orange colour, so it does not blend into the nature/sky, again with the added benefit of being able to easily track our flight path. Finally, to guarantee a successful retrieval of the CanSat, we have added a remotely-activatable buzzer which will make a loud sound when activated.

3.5 Ground station support and antenna design

The antenna design was one of our most discussed CanSat components, mainly because we wanted to design it ourselves. To achieve this, we decided it would be wise to purchase an antenna for the CanSat, instead focusing on designing the ground station antenna, which will be the topic of the following subsections.

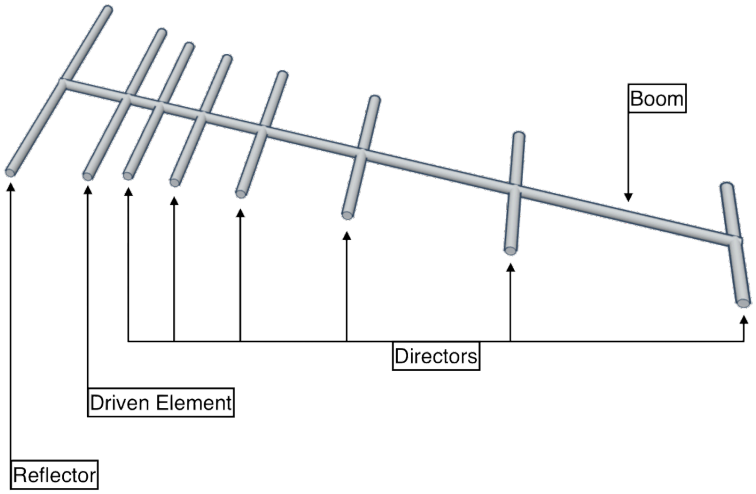
3.5.1 Preliminary design process

Back in December, when we wrote the PDR, we had pre-determined that we would use the Yagi-Uda antenna for the ground station. However, when the team re-visited the plans in January and February, equipped with the new knowledge about antennas that we had learned over the holidays, we decided on two major changes. Firstly, we would make a ground station antenna ourselves instead of buying one off the Internet. This allowed us to design the antenna with more flexibility and discretion. The second change was there were now two possible choices for our ground antenna (instead of one)—the aforementioned Yagi-Uda antenna, and a monopole quarter-wave antenna. The team ultimately decided to create a Yagi-Uda antenna, with a monopole as our backup plan. The following sections will detail our design process and considerations when making the Yagi antenna.
Both designs are engineered around our frequency — 433 MHz, the operational frequency for our CanSat components. From this, we can work out the wavelength as 69.236 cm (to 3dp) from the formula $c = f\lambda$.

3.5.2 Yagi-Uda antenna

Concept

The Yagi-Uda antenna is a really common household antenna, often seen on top of roofs to receive television signals. Typical designs have operational frequencies ranging from 30 MHz to 3 GHz, within our target frequency. This antenna is classified as a half-length antenna, meaning that the driven element (the part that carries current) length should be half of the wavelength.



As seen above, the Yagi-Uda antenna has multiple components, and has both active (connected to current) and parasitic (not connected to current) elements. From left to right:

Component	Use	Theoretical Length (cm)
Reflector	Reflects any waves going backwards and helps with direction. Parasitic.	0.55λ
Driven Element/Dipole	The main active element, which provides the current required to radiate signals to our CanSat.	0.5λ (34.64)
Directors	These help direct the signal precisely. The more directors are added, the more precise (enhancing the antenna's gain), however if the number of directors exceeds a certain amount it would have a negative effect on the impedance of the antenna. We will have between 4 and 6 directors (see below & testing plan) Parasitic.	0.45λ, decrease by 0.05λ every director (0.4λ, 0.35λ…)
Boom	The only non-radiating part of an antenna. Usually made of non-conductive material and is used to support the weight of components.	Mentioned below

Distances between components (combined will make the length of boom)
Reflector & Dipole: 0.35λ
Dipole and Director 1: 0.125λ
Between Directors: 0.2λ
Therefore, the total length of the boom (4 directors, lower bound):

$$0.35\lambda + 0.125\lambda + 0.2\lambda * 3 = 1.075\lambda = 74.476cm$$

(6 directors, upper bound):

$$0.35\lambda + 0.125\lambda + 0.2\lambda * 5 = 1.475\lambda = 102.188cm$$

We decided to use 6 directors.

Design process

Choosing Materials

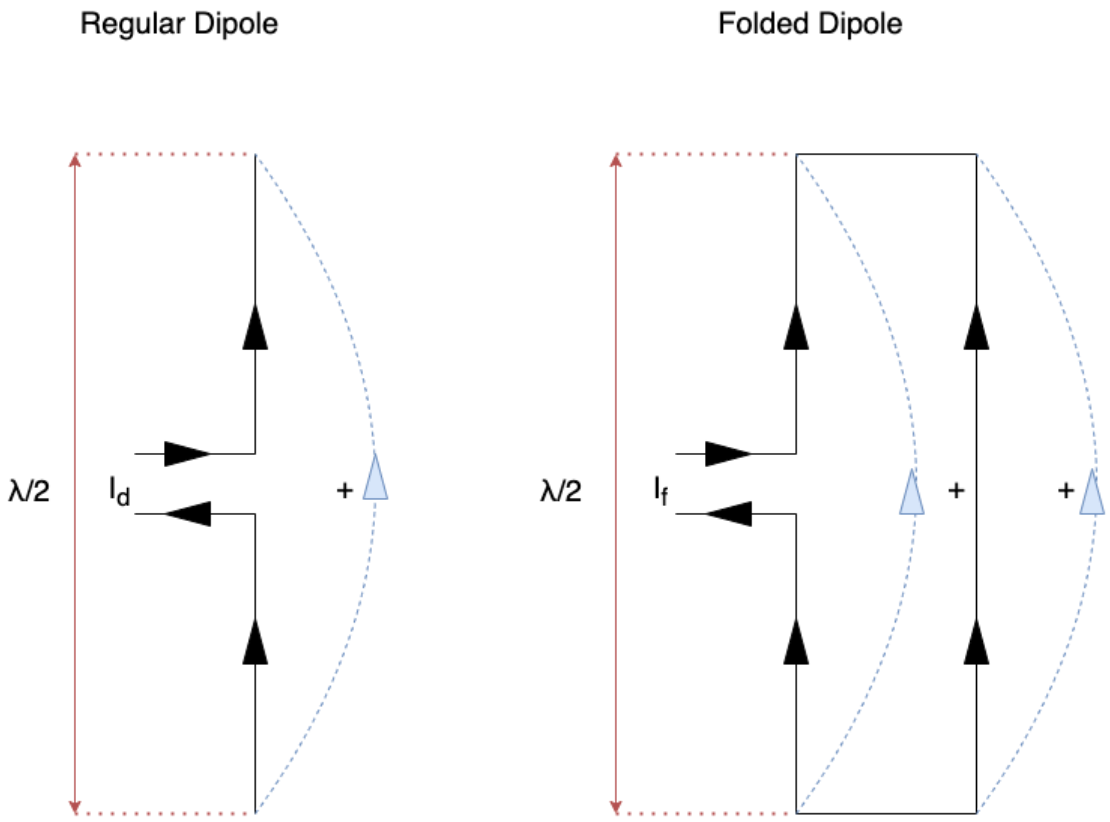
Radiating Components: The team considered multiple different possibilities of conductors, including copper, which we are using for our internal CanSat systems. However, we ultimately decided to use aluminium, as it is cheap, easy to obtain yet still conductive enough for our antenna design.
We chose a consistent diameter for all radiating components of 12mm, because this would be able to accommodate our frequency but is not too wide.
Boom: As the boom is a non-radiating component, theoretically we could use a non-metal/insulator for it. There are a couple of benefits to this:

- Using a non-metal (most likely plastic) is lightweight and in the case of PVC, more durable.
- Using an insulator would not affect the current as much as using a conductor as the boom. This is important as our calculations might be modified and we would need to also consider the boom's impact on the antenna, which would be further complicated if we had a conductor.

However, PVC plastic is one of the most harmful plastics in terms of environmental impact. To combat this, we will not dispose of spare PVC plastic, instead handing it to our school's DT or physics departments, as it might be useful for product design.
We chose PVC tubes with a diameter of 20mm, as this width would be able to support the antenna's radiating components, and would be durable enough.

Designing the driven element

The driven element is the most important component in the antenna, as it receives the signal from the source and sends it to our CanSat. The team researched multiple ways in which we could design this component, the two main ones being the regular dipole and the folded dipole.



Using the folded dipole would mean the circuit wires would be able to connect to the antenna more easily, mainly because of its design. It also has a larger gain, meaning it can transmit signals more effectively, and a more directed radiation pattern, allowing it to transmit signals more precisely. However, the impedance is far greater than using a regular dipole, so matching up with the impedance of other components would be more of a challenge.

On the other hand, although using a regular dipole is often less convenient than using a folded dipole, we chose this design. This is mainly because of the massive difference in impedance— the team had worked out a way to combat the difference on a regular dipole, not a folded dipole. The precision of the antenna will be addressed by our decision to include 6 directors, which would hopefully ensure a successful signal transmission.

Wiring the antenna

We will be using a U.FL (IPEX) cable to connect the radio sensor to the antenna.

In the model, the dipole is split into half, each with a strip of wire extended from it. One strip connects to the middle of the balun of the U.FL cable, while the other is soldered on to the side. The U.FL wire extends from there to a soldered part of the radio sensor.

Prototype & details

Dimensions:

Component	**Component Length (mm)**	**Distance from Reflector (mm)**	**Distance between previous component (mm)**
Reflector	333.72	0	N/A
Driven Element/Dipole	333.72	166.17	166

Director 1	314.54	218.09	51.93
Director 2	311.52	342.72	124.63
Director 3	308.76	491.58	148.86
Director 4	306.24	664.67	173.09
Director 5	303.94	858.53	193.86
Director 6	301.86	1066.24 (boom length)	207.71

Boom Diameter: 20mm
Radiating Component Diameter: 12mm
Gain: 10.5 dBd (12.65 dBi)
Boom Length: 1066.24 mm (106.624 cm)
Wavelength: 692.36 mm (69.236 cm)

The antenna model and a photo of our prototype can be seen in Appendix E.

The antenna should be able to send signals for up to a few kilometres, because the theoretical gain is 12.65 dBi, and a stronger gain means the signal can propagate further. The beamwidth (angle in which signal would be received) should be about 20-30 degrees.

Note: dBi is the hypothetical unit for measuring the gain of an isotropic antenna that radiates signal to all of its surroundings, and dBd is the hypothetical unit for dipoles.

Free Space Path Loss (FSPL)

One thing we need to consider about our CanSat is the fact that radio wave signals weaken over distance. We are using the upper bounds for all our variables in order to calculate the maximum signal loss. We can do that by using the FSPL formula below:

$$\mathrm{FSPL} = 20 * \log_{10} \frac{4\pi df}{c}$$

Where:

- d is distance between transmitter and receiver (km)
- c is the speed of light
- f is the frequency (Hz)
- FSPL is the Free Space Path Loss (measured in dB)

Bounds:
Frequency is 433MHz, or $4.33*10^8$ Hz,
Vertical distance (altitude) is 350m (we will adjust this for the National and European Finals)
Horizontal distance is 100m (again, will be adjusted)

We can first use Pythagoras' Theorem to calculate the maximum distance:
$\sqrt{100^2 + 350^2} \approx 364.0055m$
This can be rounded to 365m.

Then, we can plug that value and the frequency into the formula in order to get the result.

$$\mathrm{FSPL} = 20 * \log_{10} \frac{365\pi * (17.32 * 10^8)}{3 * 10^8} \approx 76.417dB$$

This data would help us identify how powerful our antennas should be and the dimensions of them. This is just a rough measurement to identify the antenna as there are many variables such as the transmitter and receiver gains, however for a CanSat this would be precise enough.

The gain of the transmitting and receiving antennas are a factor in helping to overcome FSPL, and so we decided to also calculate our antenna gains. Since we are purchasing a spring antenna for our CanSat, the product specifications listed its gain at 2.5 dBi.

From an antenna builder, we calculated that the ground antenna gain is 10.5 dB, but we can calculate maximum antenna gain by using this formula:

$$G_{dB} = 10log_{10}\left(\frac{4\pi\eta A}{\lambda^2}\right)$$

Where:

- GdB is the antenna gain (dB)
- η is the efficiency (multiply by 100)
- A is the physical aperture area (m^2)
- λ is the wavelength of the signal (m)

Ground Antenna Gain (Note that this is a hypothetical calculation but most antennas have an efficiency of near 100%):

$$10\log\left(4\pi \cdot 100 \cdot \frac{0.702192}{0.69236}\right) = 31.05333772 \approx 31.053dB$$

CanSat Antenna Gain:

$$10\log\left(4\pi \cdot 100 \cdot \frac{0.067835}{0.69236}\right) = 20.90331724 \approx 20.903dB$$

This gain will make it easier to transmit signals with success to the antenna, as this value would send strong signals for a considerable distance.

3.6 Testing

3.6.1 Mechanical testing

Carried out

Thing Being Tested	How we tested	Result	Improvement
Model Strength	By using a SUVAT equation, we calculated how far up to drop the model so it hits the ground at 8m/s (our terminal velocity) and calculated it to be 3.3m (ignoring air resistance). We then dropped our model at 0.5m and then repeated while increasing the height by 0.5m all the way to 3.5m. After each drop we took a picture and noted down any cracks or damage. The first time we did this, our surface was grass so we repeated with a new model on hard cement.	For the grass test, no cracks or damages showed up all the way until the 3.5m where some of the layers near the top were being pulled apart. For the cement test, a crack appeared at 2.5m along the base of the model	To combat the layers being pulled apart, our model designer decided to print the layers vertically so the stress is perpendicular to the printing direction. To increase the general strength of the model, we decided to commit to using polycarbonate instead of PLA and print with a high infill. After carrying out these modifications, we repeated the test and on both concrete and at grass, there was no damage until 4m.
Anemometer Section Columns	We printed only the top part of the model and repeated the model strength test mentioned before	At only 2m, on both grass and concrete, the connector snapped at the base	We widened the columns and combined with the improvements detailed in the whole model test, the columns only started to break at 4m.
Bottom to Top Connection	We attached a rope to the top of our model and started swinging it around as hard as we could, trying to simulate the G force the CanSat will face	Once the CanSat was swinging as hard as the person could swing it, the top section of the CanSat disconnected from the bottom	To combat this weakness, we have a threaded rod running through the centre of the model, holding both sections together.

	during launch/descent.	due to layer separation in the 3D prints.	
Anemometer Optimisation	Using a home-made mini wind tunnel (Appendix C), we tested different anemometers with differing cup widths and cup depths and recorded it in slow-motion. We then counted the rotations per minute.	We found the optimal cup width and cup depth to be 17.5mm and 5.8mm respectively. These gave us the highest RPM which should help us measure the speed accurately at lower wind speeds	N/A

To be carried out

Thing being tested	How we will test	Success Criteria	How to improve
Bottom to top connection	We will use the premise of our previous bottom to top connection test but this time we will have an IMU inside the model to precisely see what acceleration it is hitting	Both halves of the model stay together at least until 30g of force	Increase number of 3D printed columns connecting two halves and/or increase central rod size
Anemometer Durability	By printing the anemometer in its casing and dropping at intervals of 0.5m all the way up to 4m on both concrete and grass	No major damage should be seen until 3.5m+	Add a thin layer of foam padding just below and above the anemometer. Make the anemometer thicker
Internals Durability	After making the model as strong as we can, we'll add all the electronics inside in their proper place. We'll start dropping on grass at intervals of 0.2m all the way up to 0.4m and will stop at any sign of damage to the electronics	No damage should occur during the test	Add foam padding to the bottom of the model

3.6.2 Electrical testing

Carried Out

Thing Being Tested	How we tested	Result	Success?
Pressure Sensor	Put the sensor in a vacuum machine. Decrease the pressure by creating a vacuum and measure actual pressure with two proper pressure sensors	Measured values had deviated from the actual sensors by less than 1.5%	Yes
GPS	At one of the members' houses, we used the GPS and recorded the coordinates. We then compared these readings with the exact coordinates of the house according to Google maps and noted the offset.	We found the offset to be of a relatively small and a graph of our findings can be found here: https://cansatathena.com/wp-content/uploads/2024/02/GPS-3d-graph.gif	Yes
Temperature sensor	Placed the temperature sensor in rooms with varying temperatures and used two proper temperature sensors to get actual values	Measured values deviated by less than 1.3% from the actual values	Yes

To be Carried Out

Thing Being Tested	How we will test	Success Criteria	Improvement
Transceiver	Send test CSV files from one transceiver to another while increasing the distance in between by 50 m.	There should be minimal data loss/corruption up until 500m.	Buy a better transceiver (research required)
Light dependent resistor	Use a torch to make the LDR experience different light levels. Measure actual values using two proper light sensors.	Measured values should deviate less than 1.5% from the actual values.	Buy a better sensor (research required)
Humidity sensor	Use a water sprayer to spray water into the surrounding air, and measure with our humidity sensor. Use two proper humidity sensors to get accurate values.	Measured values should deviate less than 1.5% from the actual values.	Buy a better sensor (research required)

3.6.3 Software testing

We have tested the ground station software by creating model CSV files and seeing if the table gets updated correctly. For more intricate sub-programs like the time separation value calculator, we used a trace table to guarantee it was working. We then tried to find errors by having CSV files with missing values or wrong data-type, etc. and the script held up due to failsafes such as data type affirmation and checks for missing values.
To test the data reading on the Pico, we will take a lot of readings and see the CSV data output and see if it is what we expected. This will be done after all the sensors have been tested for accuracy.

To see if C++ was the better choice compared to Python, we made a mini-speed test. We created an infinite loop which increased a counter by 1 each time it ran and stopped the loop after 1 second. Our results showed that the C++ was over 50x faster. Though this did not account for more complex programs, it was a good benchmark for testing speed. Results can be seen in graph form as Appendix D.

3.6.4 Landing & recovery testing

Due to delivery issues, the parachute fabric has not arrived yet so we could not carry out all of our planned tests.

Carried out

Thing to Test	How we tested	Results	Improvements
Parachute cord strength	We cut a 30 cm section of the rope and tied on end to a high up bar and the other end attached to weights	The cord showed no signs of tear even after team members hung on it	N/A

To be carried out

Thing to Test	How will we test	Success Criteria	Improvement
Parachute connection	Attach the parachute cord to the parachute and the other end to the top part of the model (not printing the rest) and then attach an object which weighs 50N to the model	Everything stays together with no signs of damage	Sew the cord into the parachute more tightly
Parachute deploying	Drop an empty model with the parachute folded up from the	Parachute deploys quickly and it takes just over height of	Increase/decrease surface area

	top of our school building	building/8 seconds to land	Change folding arrangement

3.6.5 Ground station & antenna testing

Thing Being Tested	How to test	Success Criteria	How to Improve?
Impedance (Double Check)	Resistance (Resistor) + Reactance: Capacitance (multimeter) + Inductance (Need to Research)	N/A. Just a measurement.	To match impedance: use and make sure U.FL is wired properly. Alternatively gamma matching
Beamwidth	From a constant distance, place the CanSat at angles going up degrees of 5 to the antenna, from -30 - 30. Alternatively rotate the CanSat instead	Just a measurement, but current estimates based on the hypothetical gain place the range at 20-30 degrees.	Potentially reduce the number of directors(However, this would interfere with the gain so might not be appropriate)
Signal Travel Capability (Distance)	Place Antenna at a certain distance to the CanSat. Run an algorithm to receive signals from the CanSat for 5 seconds, then move to a distance 5 metres away every time, going up by a scale factor(10, then 20, then if needed 50 and 100).	The antenna should be able to receive signals from further than 365m, which is the theoretical maximum distance our CanSat would be from the ground station, if not further (preferably up to 1km) Estimates put the gain at 10.5 dBi (or 12.65 dBd).	Increase number of directors for a larger gain.

Issue	Potential Solution
Impedance issues leading to Failed Signal Transmission	Gamma matching—running a short rod parallel to the dipole to bridge the impedance difference, or changing cable width
Antenna beamwidth not enough to cover a fixed antenna	Rotating antenna—detects where the CanSat is and rotates?

3.6.6 Overall testing for launch

Cross-departmental testing

In addition to the checks mentioned above, we also need to conduct a few cross-departmental tests before our CanSat would be ready for launch.

Thing being tested	How to test	Success criteria	How to improve
Wind effect on CanSat	On a windy day, we will do the drop test and deploy the parachute mid-air, and investigate how far the parachute deviates from the initial predicted landing position. We will then calculate the predicted deviating distance for the altitude of launch day.	N/A, just a measure. Ideal scenario would be minimal wind effect on CanSat (i.e. The CanSat would not be too off-course due to wind)	If the deviation distance is too significant, we would possibly consider changing the design of the parachute or add a spill hole to our CanSat.
Leg system & parachute opening manoeuvre	We will be dropping the CanSat from a height, where the legs will be extended and the parachute will be deployed (hopefully with success). The CanSat will proceed to land.	Legs and parachute successfully deployed and CanSat lands upright.	We might look into improving the parachute design to make it more/less air resistant. We will do this by either making the area greater or adding holes.
Sending and receiving real	We will be re-enacting part of	The data of the external	We will do a full check of the

data from the CanSat.	our secondary mission (after the CanSat landing), transmitting data to the ground station every second. We will use verifiable data from other external sensors next to the CanSat to ensure correct data.	sensors matches up to the CanSat reading; the CanSat successfully transmits signals every second/otherwise instructed.	antennas of both the CanSat and the ground station to ensure all wires are soldered and connected properly. If that is not the problem, then we would improve the ground antenna design to address the specific problem.
CanSat force withstanding capability (20G)	We would attach a string to the CanSat, and receive readings from the accelerometer as we swing the CanSat around, reducing/increasing acceleration when necessary.	The CanSat does not break even beyond 20G of force exerted on it.	Improve durability of inner components and evaluate the possibility of using structural supports and more structural connectors/slot-ins for inner parts.

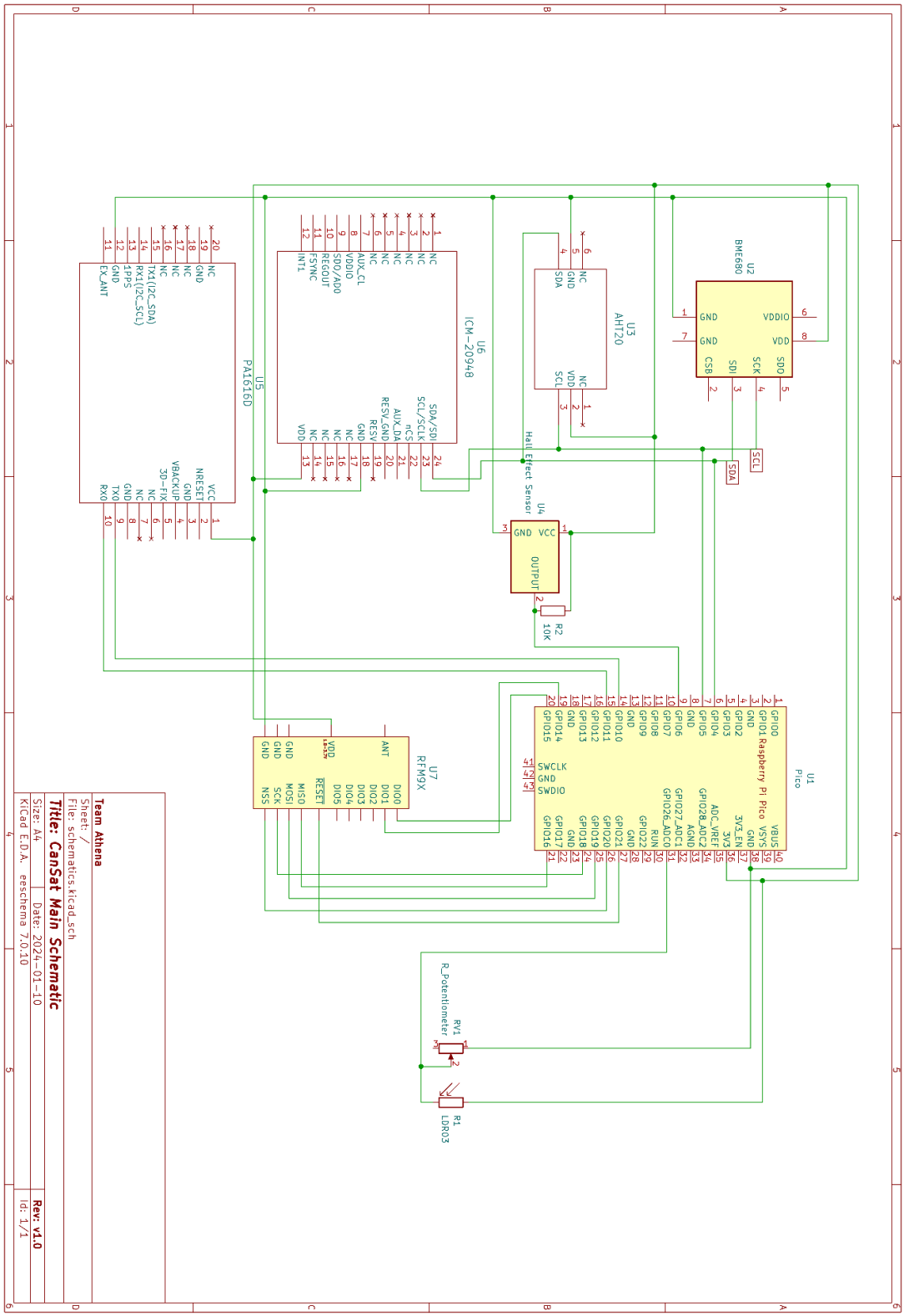
Launch simulation

To simulate launch conditions, the team currently has a plan to borrow one of the physics department drones/planes. This drone/plane will take our CanSat (or at least a model with similar weight and design as the sensors we have are expensive) and drop it from a height of around 200m in the air. As our school has a large football field, we can use it to test our CanSat. This will give us a fairly accurate test run of both our primary and secondary missions at once.

Part 4: Outreach

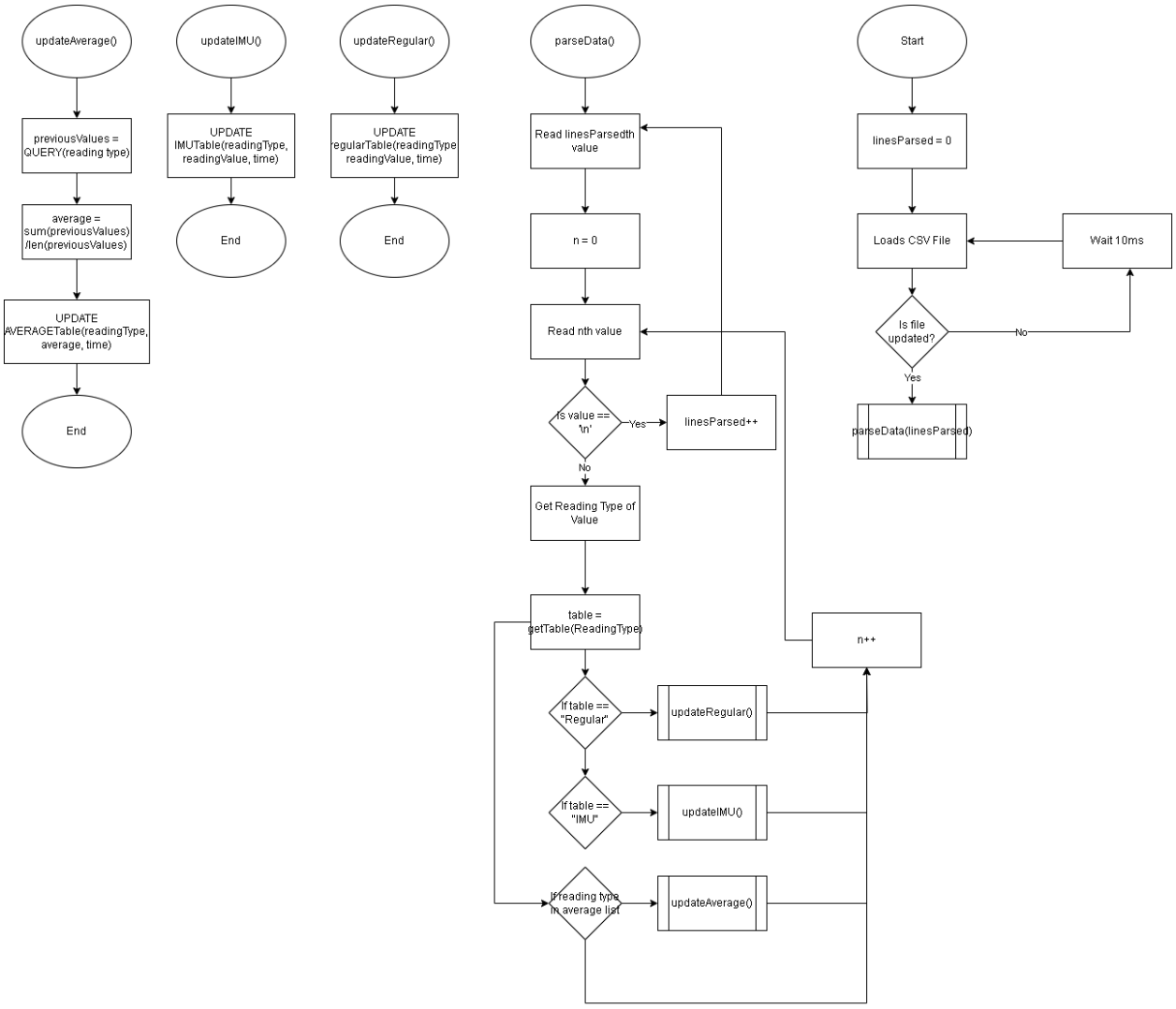
During our mission to educate the wider public about our secondary mission, we designed our own website (https://cansatathena.com) and we attracted thousands of views from over 20 different countries, with articles being regularly published daily. Article topics ranged from teamwork and perseverance to landing mechanisms and parachutes. This contrast ensured that there would be a wider engagement from a wide variety of viewers, with several pieces of feedback being given in the comments section, with constructive criticism, and praising the works. We also gained a much larger fanbase on Instagram and set up our Reddit account, which has gained much traction with viewers of the GCSE age demographic, as we regularly reply to all comments. This creates a much stronger following and we especially enjoy posting memes on our social media pages. We have also achieved our aim of attaining publicity within our school newsletter, in which we managed to educate those in our school of our secondary mission and how we plan to achieve it. We have managed to spread our influence firstly through family and friends, and then our connections have branched out from there, and this can be seen from viewership in North America, Asia, Africa, Oceania, and Europe. We also managed to attain viewership from articles in the Yahoo News, with their Young Reporters Initiative, also being on the This is Local London website (https://thisislocallondon.co.uk/news/24095861.young-reporter-tiny-team-cosmic-triumphs-luca-lo-presti-sgs). We also spread awareness by going into forms in the school, and educating them on what is CanSat, how to volunteer and to check out our website and social media pages. An example of this was when we went to forms in lower years to explain our key concepts to the CanSat, with our aim to leave them not confused about the harder concepts, but to grasp the core of our missions. In particular, the secondary mission was very relatable for the students, as they had been studying about renewable energy sources in their geography classes. We managed to give them both, the interest, and the knowledge, encouraging them to participate in future CanSat competitions. Our short-term content was to gauge interest and begin to populate support for the project, for example, the memes posted on Reddit. We also managed to time-lapse our 3D printing and this gained a lot of traction with our viewers, as time-lapsing is a trending theme in social media currently. We have successfully achieved our goal of attracting not only 13-18 year olds, but also adults have taken interest and are intrigued by the work that we have been conducting and this gave us positive reinforcement that we would be able to gauge a broad viewer base, and that our outreach programme was engaging to all audiences. Our future steps are to get articles published in the school Geography and Physics magazine about our project and mission. We will also carry on with the posting of content to social media and our website.

Appendix A



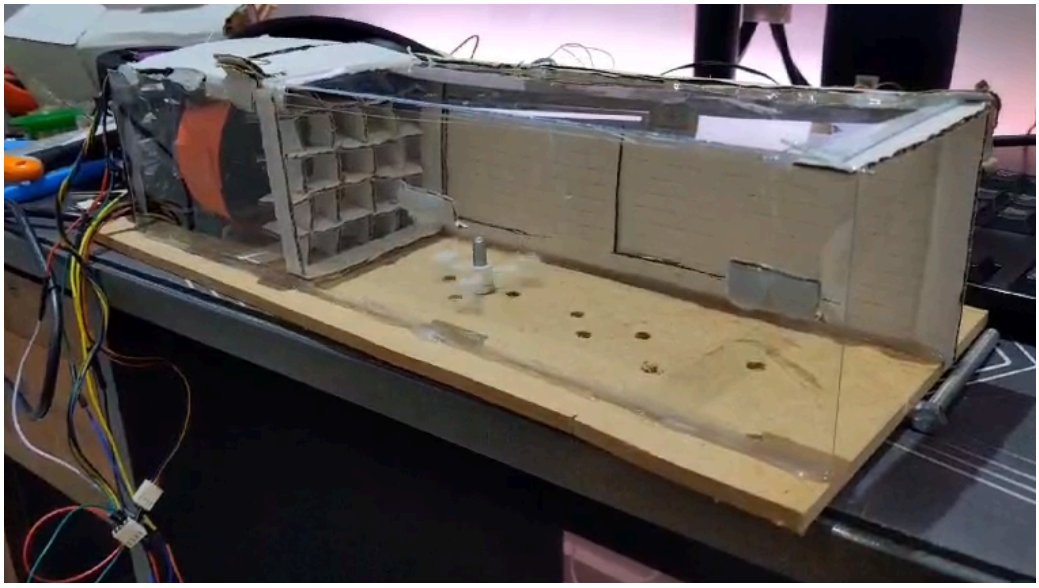
An enlarged version is available at https://cansatathena.com/wp-content/uploads/2024/02/main-schematics.png

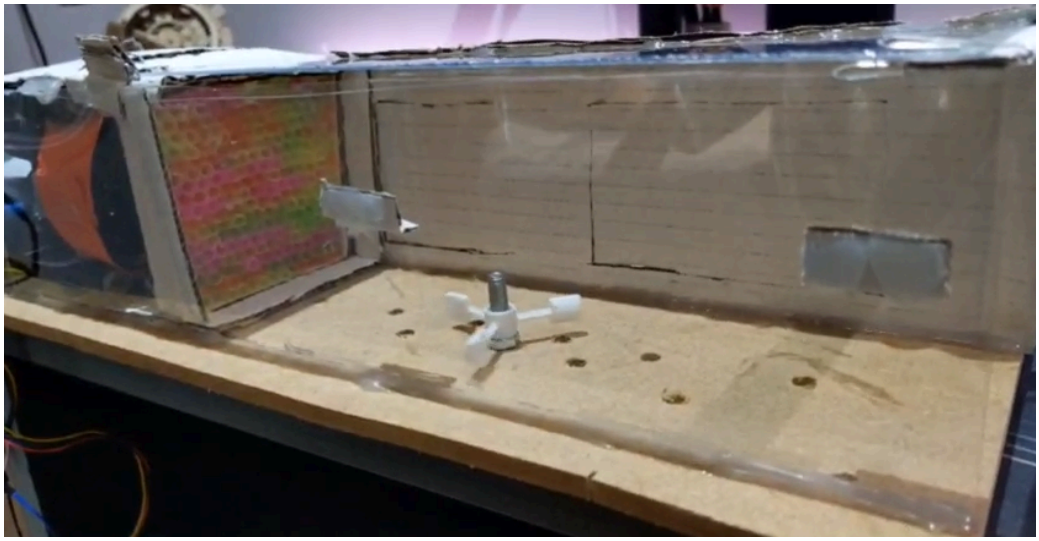
Appendix B



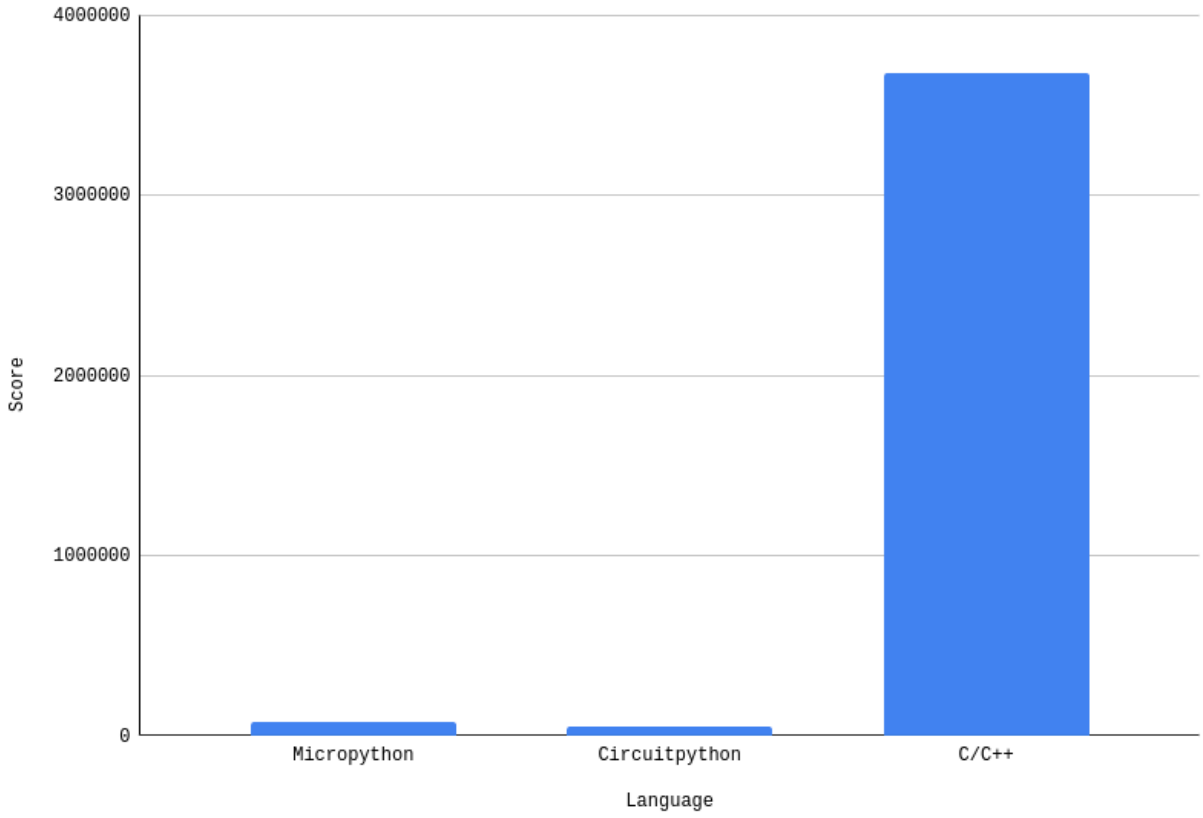
An enlarged version is available at https://cansatathena.com/wp-content/uploads/2024/02/ParseData.drawio-1.png

Appendix C





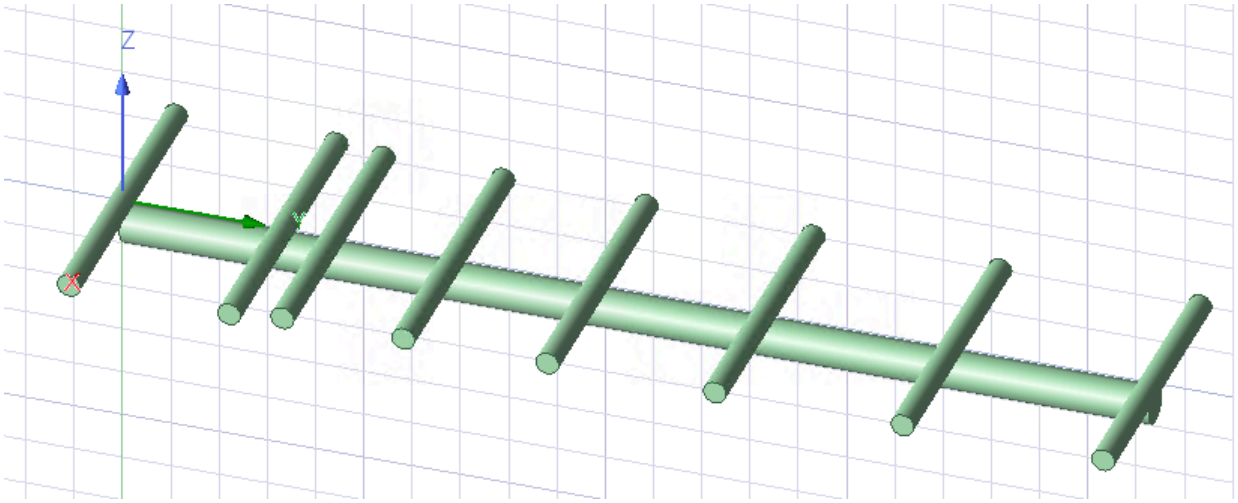
Appendix D



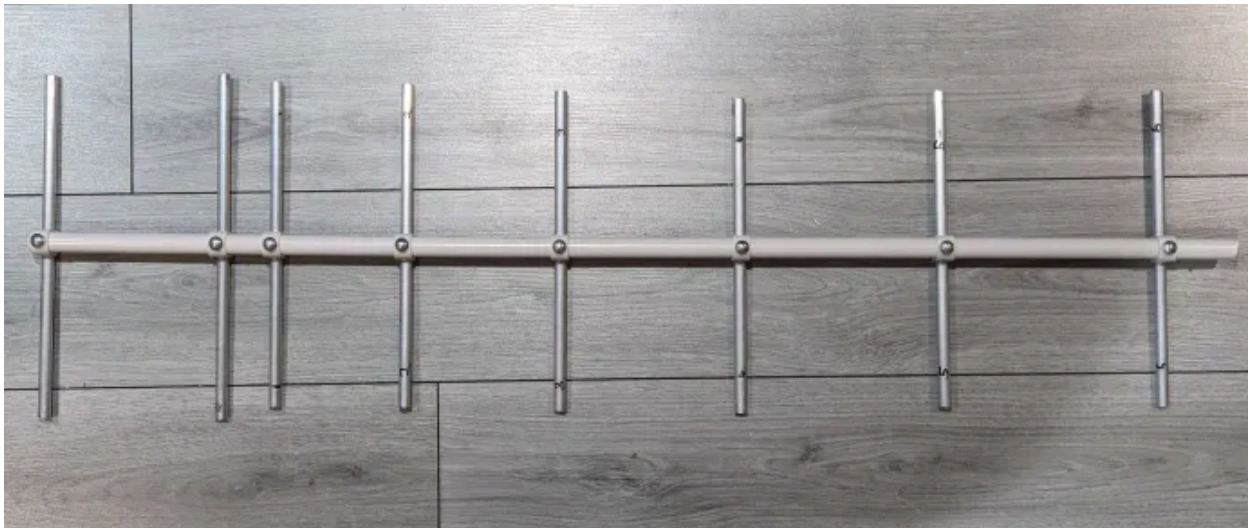
For more info, see https://cansatathena.com/2024/01/25/why-we-are-using-c-over-micropython-for-our-cansat and https://github.com/CanSat-Athena/pico-benchmark

Appendix E

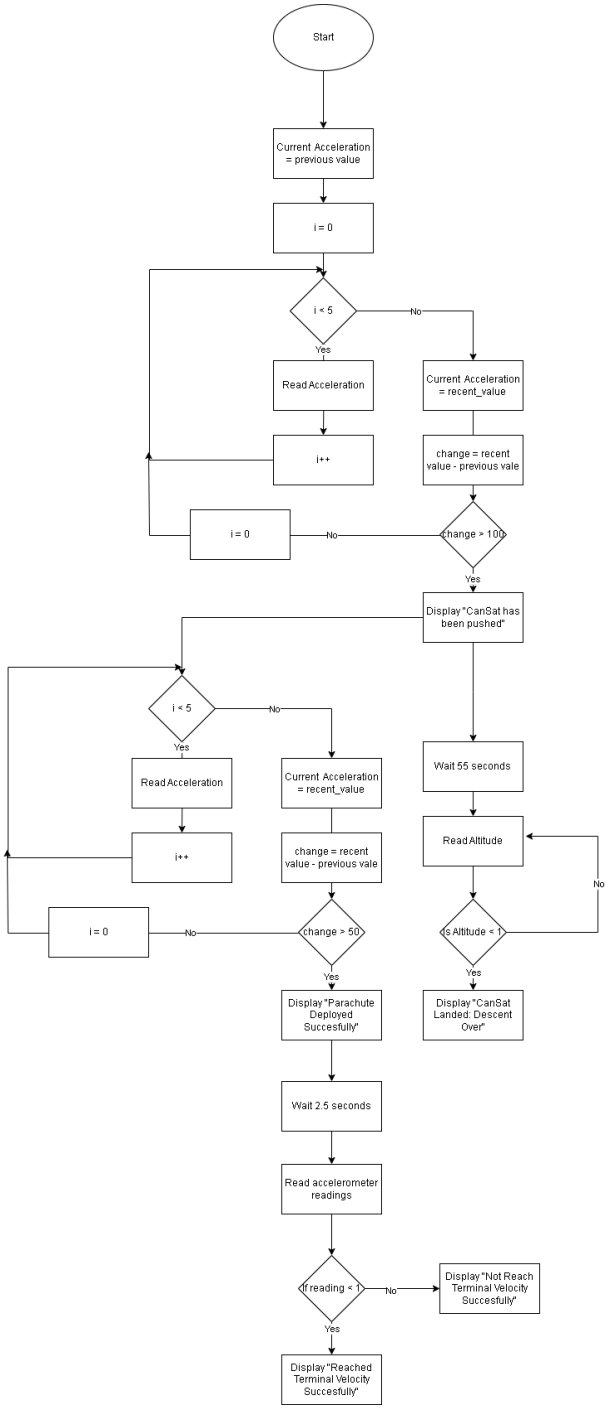
Antenna model



Our design

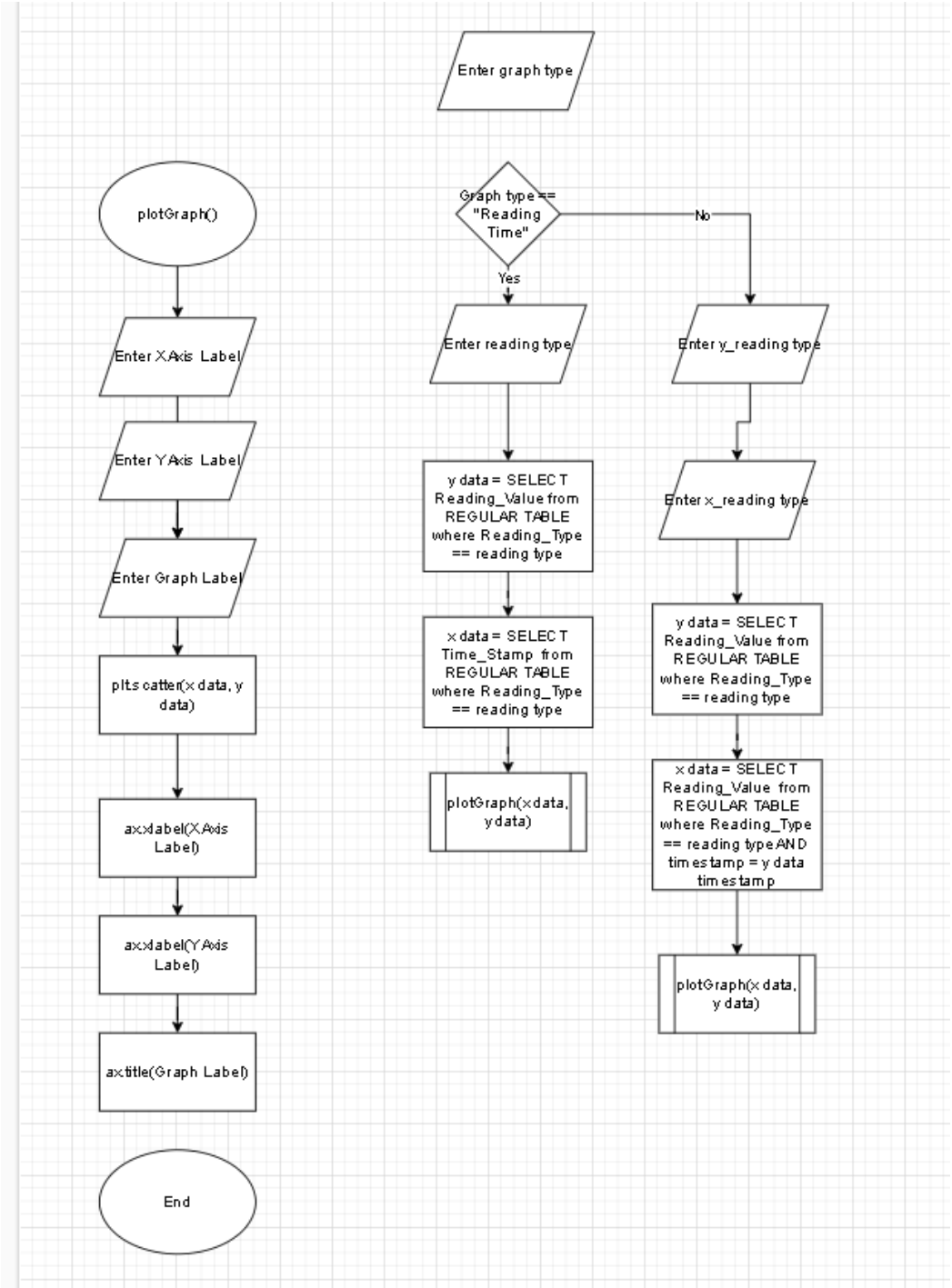


Appendix F



An enlarged version is available at: http://cansatathena.com/wp-content/uploads/2024/02/progressCheck_2.png

Appendix G



An enlarged version is available at: http://cansatathena.com/wp-content/uploads/2024/02/image-10.png

Appendix H

Component	Type	Use	Dimensions (mm)	Included?	Price
Pico LiPo (16MB)	Microcontroller	Controls everything, variant of Raspberry Pi Pico with more flash + built-in LiPo controller	51 x 21 x 8mm	No	£13.50
Adafruit BME680	Sensor	Will be used for sensing pressure and VOCs	17.8 x 20.3 x 3.0mm	No	£18.90
1200mAh 3.7V LiPo Battery	Battery	Powers everything	62 x 35 x 5mm	No	£8.00
Adafruit PA1616D	Sensor	GPS sensor to allow us to locate the CanSat and track its trajectory	25 x 35 x 6.5mm	No	£29.80
RFM96W LoRa 433MHz (Transceiver)	Radio	Transmits signals back	29 x 25 x 4mm	Yes	£19.50
RFM96W LoRa 433MHz (Transceiver)	Radio	Same as previous, but this time will be used to receive signals	29 x 25 x 4mm	Yes	£19.50
Potentiometer	Resistor	To create a voltage divider in series with the Light Dependent Resistor	-	No	£0.80
JST PH Connector Kit	Other	Better than soldering/using header pins because less space, more secure + easier to swap out	-	No	£4.30
DHT-20	Sensor	Measures the temperature and humidity (higher accuracy than BME680)	5.8 x 12.5 x 16mm	No	£4.50
ICM-20948	Sensor	Accelerometer/Gyroscope/Magnetometer	25 x 17.8 x 2mm	No	£14.70
Light dependent resistor	Sensor	Used for measuring light intensity	-	No	£2.00
Hall Effect Sensor	Sensor	Used for checking if a magnet has passed it	-	No	£5.65
TOTAL	£102.15	**Remaining Budget**	£247.85		

Appendix I

WORK CYCLE/TASK NAME														
Fortnight 1	15	16	17	18	19	20	21	Jan 5	6	7	8	9	10	11
Work on CDR Sections 1 and 2														
Advertise and Produce Content for Website, Instagram and Reddit														
Look into parachute design														
Design and Print Leg System Prototype														
Write code for the ground station														
Learn the topics required for the antenna design														
Fortnight 2	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Update and print the CanSat model														
Begin Drop Tests for CanSat														
Redesign Website UI														
Write Code for Gyroscope and Light Sensor														
Develop a Computer Model of the Antenna														
Purchase Nuts and Bolts from B&Q														

Fortnight 3	26	27	28	29	30	31	Feb 1	2	3	4	5	6	7	8
Fix Gantt Chart and format + finalise the CDR														
Purchase Parts and Build Antenna														
Website Article Production														
Write Code for Hall-Effect Sensor and Filesystem														
Finalise & Print CanSat Model														
Finalise Testing Plan Pre-launch														

WORK CYCLE	TASK NAME	START DATE	WORK PERIOD	END DATE	DURATION* (WORK DAYS)	TEAM MEMBER	PERCENT COMPLETE
First Fortnight							
	Finish CDR Sections 1 and 2	15	15-20; 5-11	20	6	RF, AH, NS	100%
	Advertise and Produce Content for Website, Instagram and Reddit	7	15-20; 5-11	11	5	LLP, NS, AH	100%
	Look into parachute design	5	15-20; 5-11	11	7	NP	100%
	Design and Print Leg System Prototype	5	15-20; 5-11	11	7	AEC	100%
	Write code for the ground station	15	15-20; 5-11	21	7	AEC, NP	100%
	Learn the topics required for the antenna design	5	15-20; 5-11	11	7	RF	100%
Second Fortnight							
	Update and print the CanSat model	14	12-25	20	7	NS, LLP	100%
	Begin Drop Tests for CanSat	21	12-25	24	4	AH, NS	100%

	Redesign Website UI	12	12-25	13	2	RF, LLP	100%
	Write Code for Gyroscope and Light Sensor	12	12-25	20	9	AEC, NP	100%
	Develop a Computer Model of the Antenna	17	12-25	25	9	RF, AEC	100%
	Purchase Nuts and Bolts from B&Q	22	12-25	22	1	NP, RF	100%
Third Fortnight							
	Fix Gantt Chart and format + finalise the CDR	26	26-8	8	14	NS, RF, AH	100%
	Purchase Parts and Build Antenna	26	26-8	31	6	AEC, NP, RF	100%
	Website Article Production	29	26-8	2	5	AH, NS, LLP	100%
	Write Code for Hall-Effect Sensor and Filesystem	1	26-8	8	8	NP, AEC	100%
	Finalise & Print CanSat Model	4	26-8	8	5	AEC lead, whole team	100%
	Finalise Testing Plan Pre-launch	6	26-8	8	3	NP and RF lead, w. team	100%

Appendix J

To analyse data from our secondary mission, we will try to determine if the launchsite is suitable for renewable energy. However our analysis won't be that accurate since we will only be reading data for a few minutes unlike the months of data needed to get an accurate analysis. To see if the launch site is suitable for wind power, we will use the anemometer data. This tells us the wind speed however wind speed changes with height so we can use the logarithmic wind shear formula to estimate the wind speed at 90 m (roughly how high an average wind turbine is). The formula can be seen below:

$$v_2 = v_1 \frac{\ln\left(\frac{h_2}{z_0}\right)}{\ln\left(\frac{h_1}{z_0}\right)}$$

V_1 = Velocity at height h_1 (How far the anemometer is off the ground)
V_2 = Velocity at height h_2 (Assume height of a turbine is 90m)
Z_0 = Roughness (See table below)

Roughness class	Roughness length z_0	Land cover types
0	0.0002 m	Water surfaces: seas and Lakes
0.5	0.0024 m	Open terrain with smooth surface, e.g. concrete, airport runways, mown grass etc.
1	0.03 m	Open agricultural land without fences and hedges; maybe some far apart buildings and very gentle hills
1.5	0.055 m	Agricultural land with a few buildings and 8 m high hedges seperated by more than 1 km
2	0.1 m	Agricultural land with a few buildings and 8 m high hedges seperated by approx. 500 m
2.5	0.2 m	Agricultural land with many trees, bushes and plants, or 8 m high hedges seperated by approx. 250 m
3	0.4 m	Towns, villages, agricultural land with many or high hedges, forests and very rough and uneven terrain
3.5	0.6 m	Large towns with high buildings
4	1.6 m	Large cities with high buildings and skyscrapers

Now we have the wind speed at 90m, we can now compare the wind speed with the wind speed requirements of a turbine. We have found a turbine needs at least 5m/s of wind to work and at 25m/s the turbine starts breaking. We'll first make sure the speed is within this range. We have also found that the best wind speed for peak efficiency is 15m/s so the closer the measured wind speed is to this, the more likely the area is good for wind speed. Another thing we have to look out for is how long the wind turbine will turn as a good wind turbine needs to turn at least 90% of the time however due to our short readings time, we cannot carry out this analysis. To check for Solar power we will use the LDR data. We can then figure out the lux value of that area then we can convert this into Solar Irradiance by multiplying the value by 0.0079 (assuming the light is from the Sun). The top solar power generating places in the world have a Solar Irradiance of 250 -200W/m^2 per day so we will be looking for a Solar Irradiance value in this range. Finally, solar panels have an optimum temperature of 25 degrees celsius so we will check if the temperature readings are near this value. Hydroelectric power is a bit difficult to check for as we don't have access to many sensors due to the size limitation however we can use humidity readings to get a rough estimate. The DHT20 measures relative humidity and when the relative humidity gets above 90%, it is most likely a sign that there will be rainfall. That's why we will be looking for 90%+ humidity and a low temperature . However, in reality we would also need to see how long does this rain last as it needs to rain consistently for water to build up but we cannot carry this out due to again our reading time limitations. Finally to check for Geothermal energy, we will be using the analysis from before if there's water around as you need to heat up the water. We will also be using the accelerometer and gyrometer to see if there are earthquakes as that's a sign of continental movement and therefore magma. Obviously we will not detect an earthquake on launch (hopefully) so we plan to place the CanSat on a platform and then move it around to demonstrate the system working.