CanSat 2023
Preliminary Design Review (PDR)

VarSat1

Team name: VarSat 1

Country: Poland

Table of Contents

Table of Figures

1. Changelog

PDR:

- We had a team members change – one of the team mates left and on his place another one joined in – Marcin Fedyna.
- We've added new satellite firmware algorithms – vertical speed, array data sending, audible and visual notifications.

2. Introduction

2.1 Team organization and roles

Our team consists of four people from VII High School in Warsaw and physics teacher, who inspects the project.

1. **Franciszek Łada**
   a) Interests: Associates his future with IT. Knows lots of things in terms of Linux and ethical hacking. Knows Arduino and embedded ecosystem well. He also likes photography.
   b) Field of work, also expected, within the team: CEO of VarSat project and its originator. Makes entire electronical design for satellite and ground station, writes satellite's firmware and project documentation. Manages project website.
   c) Hours dedicated weekly, at school, and after: ~20h.
2. **Wiktor Jędrych**
   a) Interests: Robotics, programming and electronics. Often makes projects in these fields.
   b) Field of work, also expected, within the team: 3D model design and printing, ground station program.
   c) Hours dedicated weekly, at school, and after: ~15h.
3. **Mateusz Bylica**
   a) Interests: C# and Python programming. In free time makes games for Windows and Android.
   b) Field of work, also expected, within the team: Ground station design and programming. Manages project social media and finances.
   c) Hours dedicated weekly, at school, and after: ~14h.
4. **Marcin Fedyna**
   a) Interests: He likes 3D modeling and C++ programming. He likes audio engineering and coordination.
   b) Field of work, also expected, within the team: Documentation, parachute design, some graphic designs.
   c) Hours dedicated weekly, at school, and after: ~8h
5. **Katarzyna Mazur**
   a) Coordinating teacher, project supervisor and physics teacher. A graduate of Wydział Matematyki, Fizyki i Informatyki Uniwersytetu Marii Skłodowskiej-Curie in Lublin, majoring in physics.

2.2 Mission objectives

Primary mission is to launch the CanSat to an altitude of 1-3km above ground level and to measure temperature and pressure. Satellite will both save this data into it's built-in memory and send it to the ground station via radio link. Then, it has to safely come back to the ground.

Secondary mission - the probe will mainly focus on measuring the properties of the atmosphere and flight telemetry. The goal is also to analyze the collected data, which will help to determine atmosphere quality and it's impact on life and health. After landing, the satellite will be still able to transmit data from ground for at least 12 hours.
The reason for choosing this topic is a growing interest in air pollution and it's impact on life. Additionally, the satellite could provide important information, whether desired space is suitable for colonizing.

VarSat 1 mission will be considered successful, if the following guidelines are met:

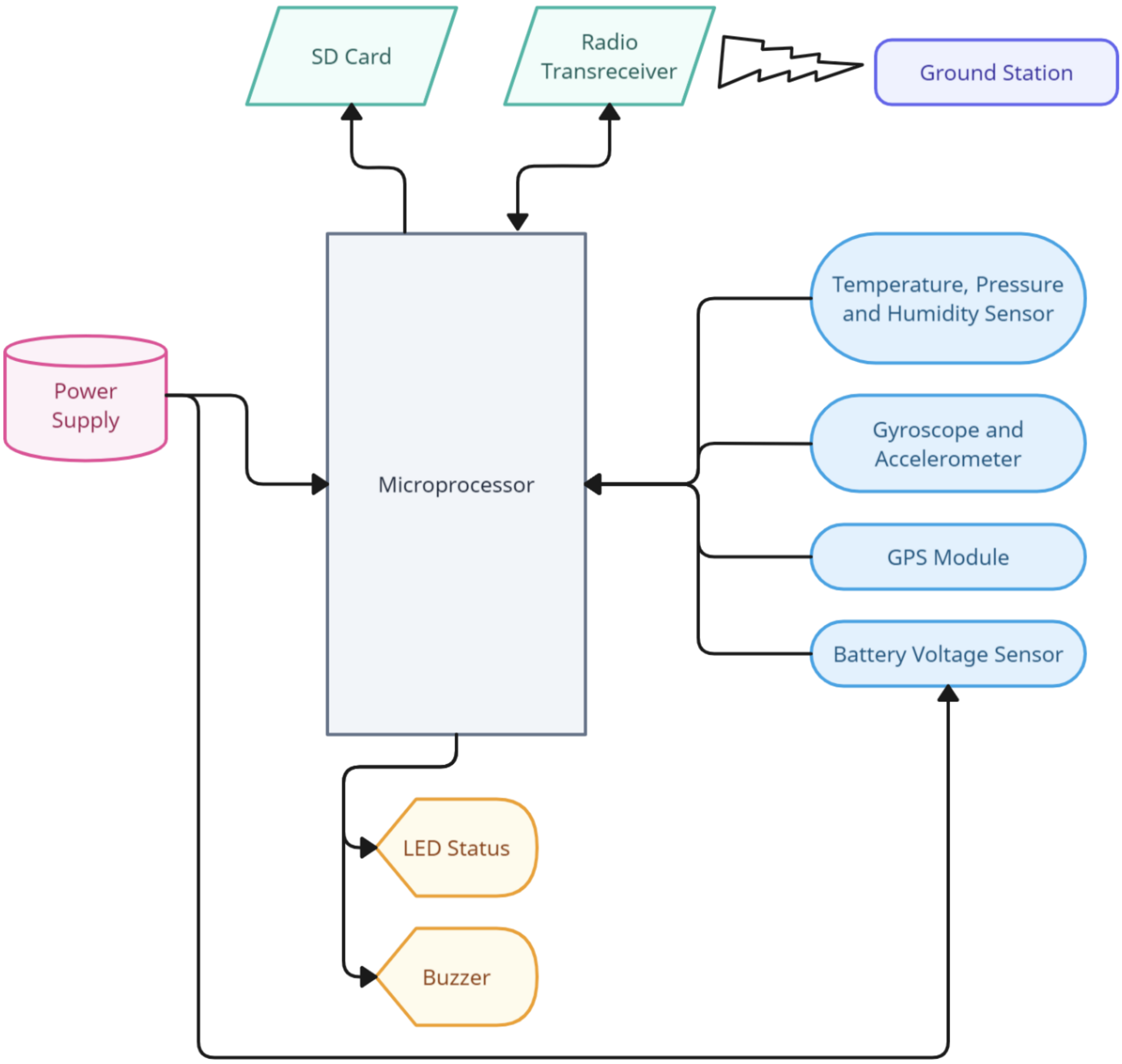
- Safe start
- Successful procedure of leaving the launch device (rocket, drone, etc.)
- Stable radio communication (stable – in these terms – without major data corruption, allowing to read necessary data, frequently enough for desired mission).
- Saving the obtained data to the satellite's built-in memory and saving logs on the ground station.
- Safe landing without severe damage (severe damage – damage that will affect CanSat's operation).
- Satellite continuing working after landing.

3. CanSat description

3.1 Mission overview

Our CanSat is designed to be launched from a rocket at an altitude of 1-3km AGL. The CanSat is going to descend no faster than 8,5m/s, during descent it'll collect all necessary data such as temperature, humidity, pressure, altitude AGL and MSL, GPS position, telemetry from 6DoF IMU (six degrees of freedom inertial measurement unit), voltages, error count and others. It will transmit important data to the ground station in 900ms intervals and simultaneously save it to the MicroSD card with 160ms intervals. Once landed, CanSat will keep working, logging, and sending data for at least 12 hours.

Functional block diagram:



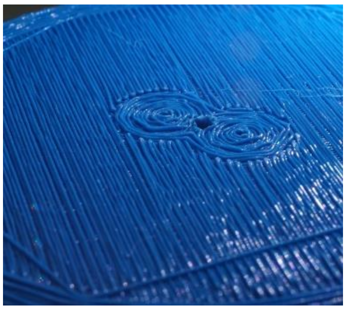
Photograph 1 - Functional Block Diagram

3.2 Mechanical & Structural Design

Our CanSat enclosure was designed in Autodesk Fusion 360 and 3D printed. We use PLA+ filament as a material - it's elastic, so will absorb landing energy, but strong enough to withstand rocket accelerations of up to 20G. It’s a cylinder with dimensions of 113mm x 65mm. The completed CanSat weighs 190g, so we added 120g of lead ballast to match the 300-350g requirement.

The main PCB is slid on rails into enclosure through its bottom opening. The enclosure also has a solid mounting for a parachute, holes for an elastic antenna, led and speaker, and also a mounting for battery and its BMS. Design is maximized for best usage of available space.

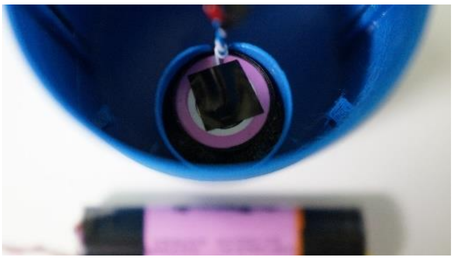
Photograph 4 - PCB Rails, Battery Holder



Photograph 3 - Antenna Hole, Parachute Mounts

Photograph 2 - Enclosure

For the battery mounting, we used one cell of 18650 li-ion 3500mAh Samsung-35E. We have secured it with foam around it, so it fits tightly and is protected from impacts, as it could result in an internal short. [see: Power supply]



Photograph 5 - Battery Mounting

Our CanSat will be descending on a own-made parachute [see: Recovery system].

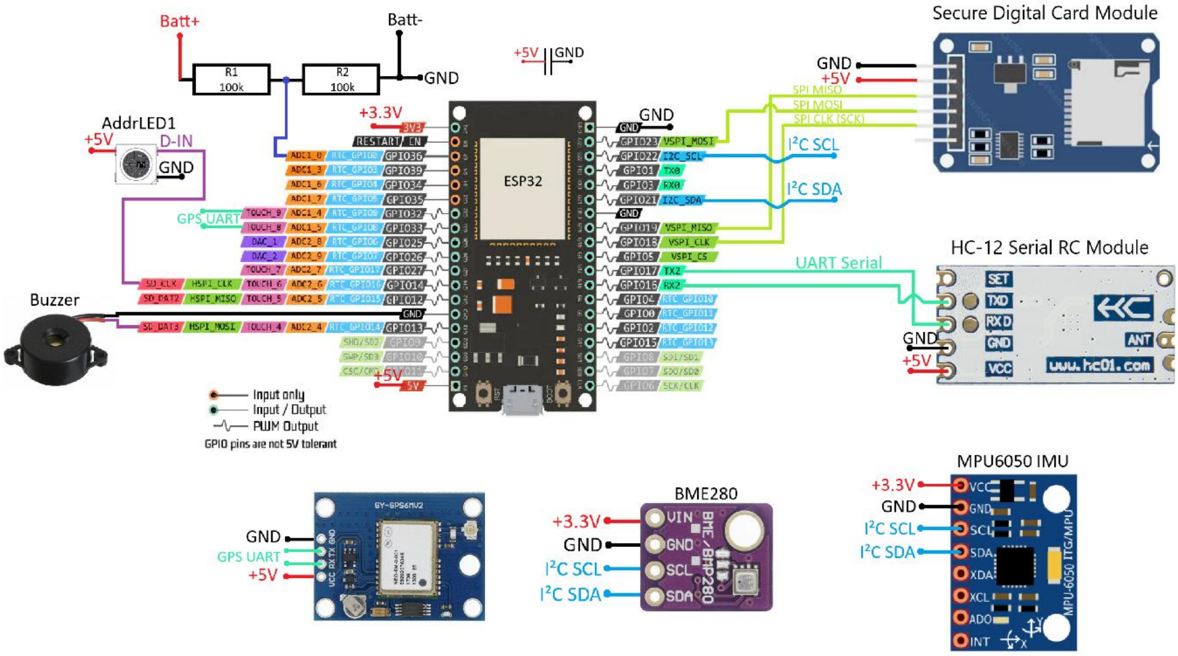
The additional 40mm x 65mm space will contain a freely packed parachute and an elastic radio antenna.

Whole project was built to be easy to use, while having lots of features.

3.3 Electrical design

3.3.1 General architecture

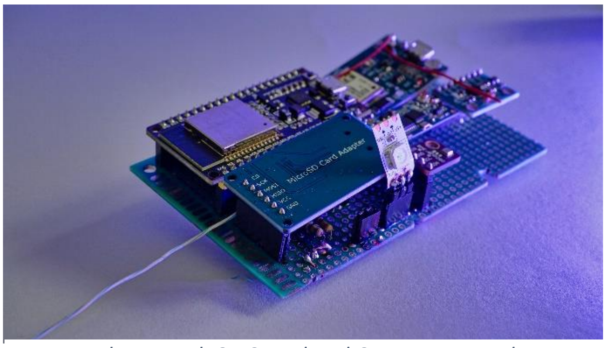
Circuit schematic:



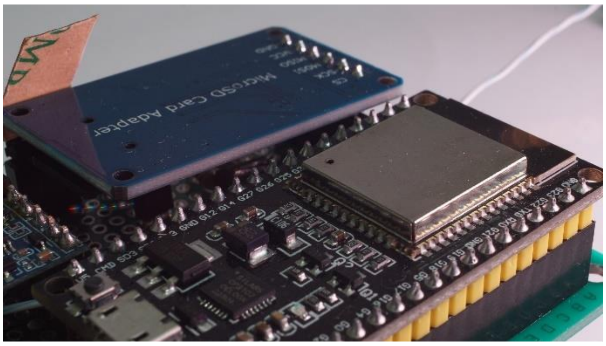
Photograph 6 - Circuit Schematic

Before we transferred the idea to the satellite, we've made lots of tests on a breadboard. After we wrote the firmware, we finally soldered components to the PCB, designed to be put in CanSat.

The PCB is a standard PCB prototyping, through-hole, one layer board. It has components mounted on goldpins for easy replacement.



Photograph 8 - Completed Computer Board



Photograph 7 - ESP32 and MicroSD

3.3.2 Primary mission devices

For the brain of the CanSat computer we chose the ESP32 microprocessor. It has 2 cores, each with 240MHz clock. It also contains 540KB of RAM, 4 MB of memory, 38 I/O pins, ADCs, DACs, PWM and much more. We chose this solution, because it has a low power consumption when wireless is turned off, whilst still having plenty of performance. It also supports FreeRTOS, which gives us ability to run multiple tasks simultaneously, with active use of both cores. For final design, we've chose ESP32 on a 38pin dev module. We both program and power it using integrated Micro USB port.

Our CanSat saves data onto the built-in Micro Secure Digital Card, connected via SPI. Data is saved with 160ms intervals, which is best for high data write rate and no data corruption. For the radio link, we used HC-12 serial radio module [see: Communication system], connected via UART to the hardware serial 1 port.

One of the primary mission objectives is to measure temperature, humidity and pressure. We achieved this using the Bosch BME280 sensor.
According to it's datasheet and real world tests, it provides high accuracy and range, and a low noise output. It is connected using the I^2C bus, together with the IMU.

3.3.3 Secondary mission devices

One of the most important goals is to measure the flight telemetry. We use the MPU6050 as the IMU, because of it's ease of use, available documentation and good properties. It can measure accelerations of up to 16G and rotation of up to 2000°/s. It also has an integrated temperature sensor, which allows us to monitor the satellite's inside temperature. We connected it to the satellite computer via the I^2C bus. Using different I^2C addresses, we can use multiple sensors on the same line.

For precise positioning we've installed the u-blox Neo-6M module. It has a 2.5m horizontal accuracy and is able to catch a cold fix within a minute and a hot fix in under 3 seconds. Unfortunately, when it arrived, the integrated ceramic antenna was faulty. But we had an idea – what if use frequency matched dipole? We soldered two wires for 1575 MHz frequency, at an 90 degree angle. And it worked. Now module sees many satellites, even indoors. Dipole also allows us to have a more omnidirectional reception pattern, which is better during flight, as the CanSat can rotate in different directions.

We didn't receive the ENS160 atmospheric gases sensos yet, but we already planned it's usage in advance.

The satellite computer is also containing a battery voltage sensor. A standard 1S li-ion battery usually provides up to 4.2V, which is too much for ESP. So, we connected it to the ESP32 analog input via a voltage divider, made from two 100k Ω resistors. It works with a real accuracy of about 0.1V, but it may vary due to ADC nonlinearity.

The computer board also has an addressable WS2812B RGB diode as a status light and a loud buzzer for playing find signal after landing. Both devices are also used for status and debug messages.

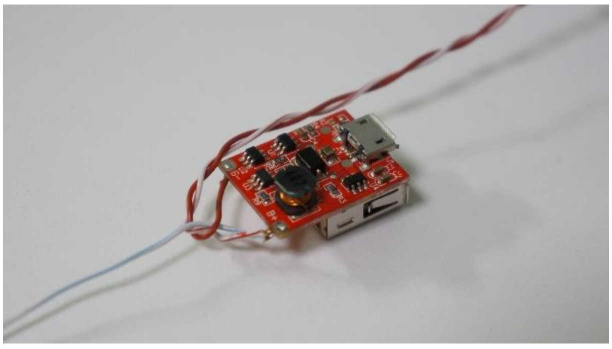
Lastly, we've added a 4700µF 10V capacitor on a 5V line, because it provides power for radio devices – HC-12 radio and a u-blox GPS. They have large current peaks of up to 1A, so it was a mandatory decision, because there were slight voltage drops.

3.3.4 Power supply

For the satellite we needed a small battery, that would provide power for a long time. This was a really important thing, so we chose 18650 batteries, as they perfectly fit into the enclosure. More precisely, we installed a single cell of Samsung 35E li-ion battery, that has a typical capacity of 3500 mAh, maximum voltage of 4.2V and 3.6V nominal. It provides up to 15h of working time for our CanSat.

Photograph 9 - Battery

Lithium Ion batteries require strict control in terms of charging and discharging cycles. It was necessary to add a BMS. We chose one with Micro USB input and USB-A output for ease of use. We connected the BMS with the CanSat computer via a short and thin USB cable. There is also a third cable going from the battery to the voltage sensor.



Photograph 11 - BMS Board

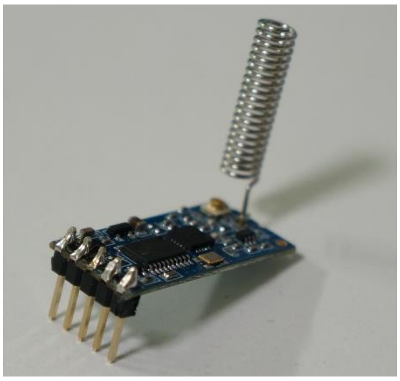
Photograph 10 - Battery Connections Protected

Running computer consumes about 220mA of power from the battery. Using this value we estimate about 15h of working time. Most of the power is consumed by the radios, which is 150-200mA. The rest are sensors and ESP32. In the future, we plan to add power saving features to extend the battery life to over 1 day.

3.5.5 Communication system

We needed to somehow communicate with the ground station. We had a strict limit of 100mW (+20dBm), so we ended up with two HC-12 modules. They work like wireless serial ports. We set them up with the 9600 baud rate. They operate at 433.4-473 MHz, with 100 channels separated with 400 KHz steps.

In future, we plan to use better antennas – a very good one on ground and a frequency matched one in the satellite. We haven't tested the wire antenna in the satellite yet.

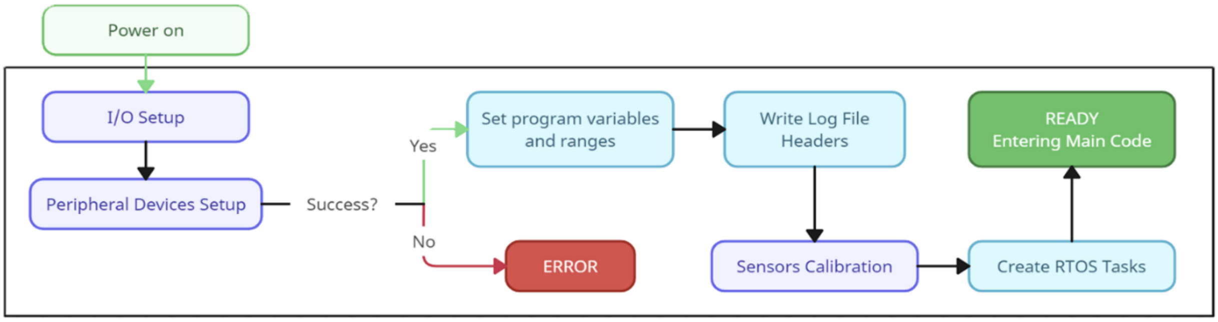


Photograph 12 - HC-12 Radio Module

3.4 Software design

Whole satellite program is written in C++ using Arduino IDE environment, because of it's development ease and full support of ESP32 platform. We used a few public libraries, for example Adafruit ones for sensors.

Firstly, after turning the computer on, the firmware enters the **Setup Stage**:

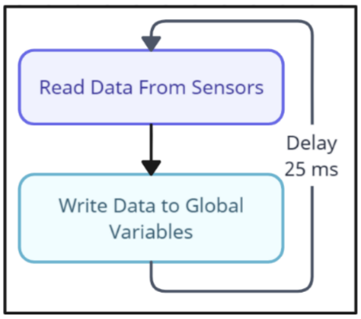


Photograph 13 - Setup Code Flow

It runs the serial and I/O setup – assigns interfaces and selects settings. Next, it initializes peripheral devices, like SD Card, IMU and environmental sensor. It uses audible and visual notifications. For example, when there is an issue with starting a sensor, speaker beeps and RGB diode becomes red. If this procedure succeeds, the firmware sets program variables and ranges, like IMU acceleration limit. After this, the headers are written to the files for easier data management. Then the firmware calibrates necessary values, like ground level. It takes 4 readings in 250ms intervals and calculates an average value, which is used to show the altitude AGL value. Then, the firmware creates 5 loop FreeRTOS tasks. It's an easy and convenient way of multitasking. If this procedure goes all right, the satellite is ready to flight.

Task 1:

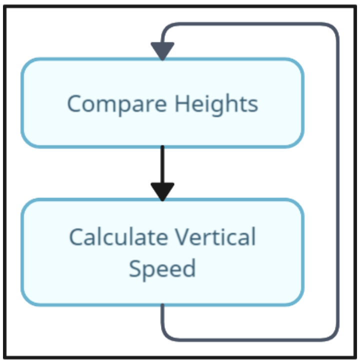
It is a simple task – using an infinite loop, code reads data from all sensors and writes values to the global variables. It is a very important task – the rest of the code relies on it. A completely separate task for reading sensors data is used to make multitasking easier.
For example, when values would be read from sensors in many different tasks, the intervals would eventually match, causing two instances of code to read from one bus or device exactly in the same time. This would crash the system and reboot it. Using a separate task prevents from it.



Photograph 14 - Task 1 Code Flow

Task 2:

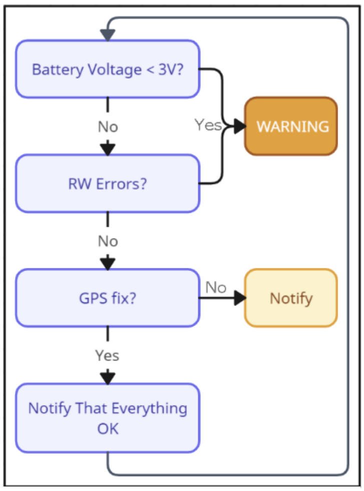
This task calculates vertical speed based on the height change.



Photograph 15 - Task 2 Code Flow

Task 3:

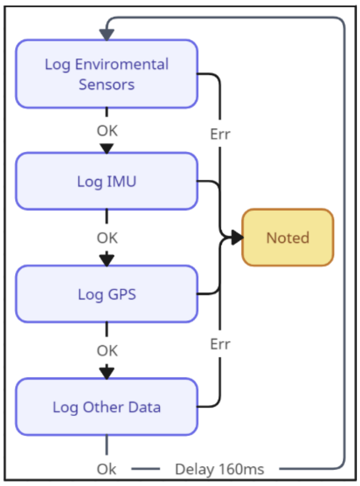
This task is used for status indication. When battery voltage drops below 3 volts or when there are significant read/write errors, the satellite gives us an audiovisual warning. It also checks whether the GPS has established a fix. If it did, we get a green LED status that everything is all right.



Photograph 16 - Task 3 Code Flow

Task 4:

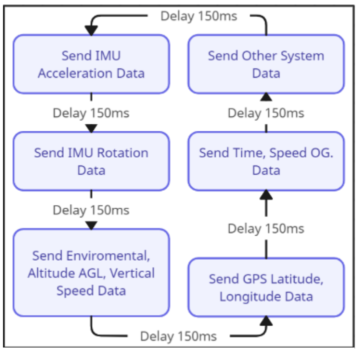
One of the more important tasks is the one that saves collected data onto the memory. It gathers data from environmental sensors, IMU, GPS and other system data to write it in 160ms intervals on the MicroSD Card. It is the best value if we want to have lots of data, that doesn't take much space.
The task also has an algorithm that checks if the write operation went correctly. If it didn't, the firmware notes this error to the value that is used to count memory errors. It gives us an important ability to determine data corruption.



Photograph 17 - Task 4 Code Flow

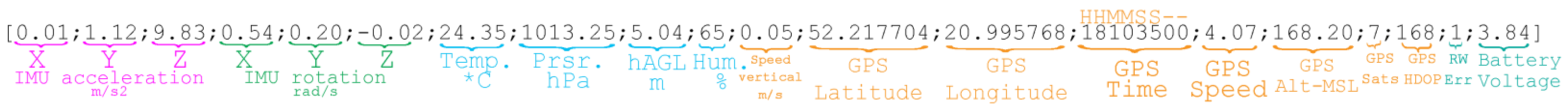
Task 5:

The last of firmware's tasks is the one that sends all necessary data to the ground station. It sends an array divided into 6 parts with 150ms intervals, to keep data from corruption.



Photograph 18 - Task 5 Code Flow

Arrays use our own data coding algorithm. Example data array:

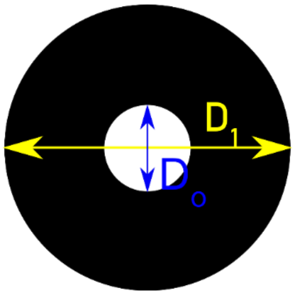


Photograph 19 - Data Array Description

3.5 Recovery system

For the CanSat recovery system we decided to use a circular, flat parachute made of ripstop nylon, which has high durability, low mass, and low cost. The parachute will be created from one piece of material and will be fixed to the CanSat body using six pieces of fishing line string, each being 70 centimeters long.

Firstly, we had to calculate the parachute's surface area based on the CanSat mass, for which we chose to use the formulas present on the NASA website. We got a value of about 1000 cm^2. For better flight stability we decided that our parachute will contain a stabilizer hole in the center.



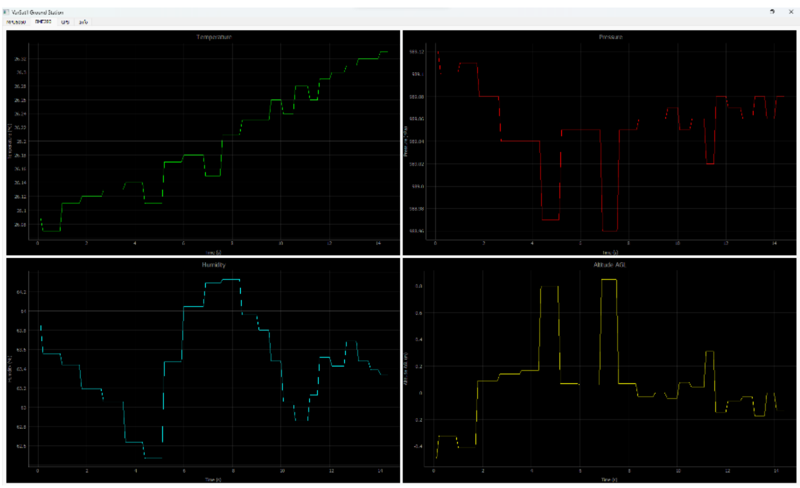
So, the final design will be a circle with a diameter of 40 cm (D_1), and a hole in it with a diameter of 9 cm (D_0).

The satellite will also have a loud buzzer, that will start to beep when the computer detects that the satellite has landed.

3.6 Ground support equipment

The ground equipment consists of a Linux laptop and a station module connected via USB. We built it with Arduino UNO and a second HC-12 radio module. Arduino is running a program that transfers data from radio module via software serial to the hardware serial port connected to the computer.

Computer runs our custom Python program, that reads and decodes the data. Then, it shows everything on graphs. It also has local data saving options. We plan to implement landing place prediction based on telemetry data, and use sockets to show graphs on multiple computers on the network.



Photograph 20 - Ground Station Program Screenshot

4. Test campaign

4.1 Primary mission tests

Our primary mission is to measure temperature and pressure, and to send and save this data. We tested the temperature and humidity sensing by placing our CanSat in different places, for example indoors, outdoors, fridge. We made sure that the saved data is good by looking at the tables and importing the data into the spreadsheet.

4.2 Secondary mission tests

Firstly, we tested the IMU. We made high resolution data saves and the sensor seems all right. Gyroscope readings are accurate, but we didn't test the accelerometer accuracy yet.
Next, we tested the GPS. After some trouble with antennas we finally got it working perfectly. We went outdoors to see it in action and we were surprised by it's performance, which was great.
Perfect working of the ground station program, radio communication and SD card was also confirmed.

Totally, on the software and hardware testing purposes we spent over 20 hours, and we didn't see any bad signs. We've even made a long 5 hour test run.

4.3 Tests of recovery system

We haven't made the parachute design yet, but we have all the necessary calculations. On the first few tests we plan to use a dummy-mass model, which will be dropped from about 30m. When everything goes as planned, we'll move to real satellite drops. Later, we want to drop it from a drone, from an altitude of even 500m AGL.

4.4 Communication system range tests

We only tested the setup once, in a worst case scenario - with stock antennas, in the center of biggest city of Poland, with a lot of interferences and without the proper power supply – and we got near 800m of range in a straight line, without data corruption, both modules on the same height, with moderate signal visibility of each, and not fully polarity-matched. With this setup, on an area with no interferences, with clear line of sight and with larger altitude difference, we predict the distance to be slightly over 1km.

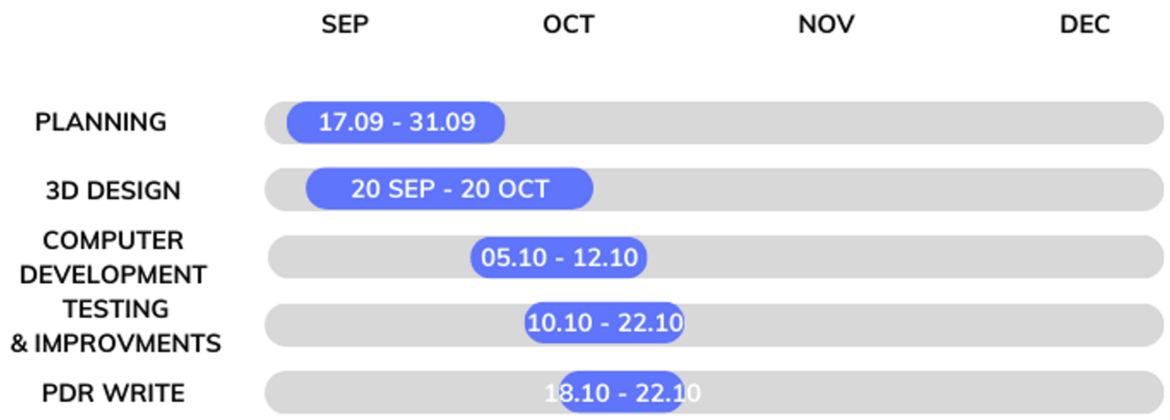
We also plan to use different antennas, which will result in a lot better transmission power and reception sensitivity. We predict the radio range to be about 2 km in good conditions.

4.5 Energy budget tests

We didn't test the CanSat battery life time from 100% to 0%, though using an average current consumption of 220mA and battery capacity of 3500mAh we predict the lifetime to be over 14h. We have to test it as soon as possible.

5. Project planning

5.1 Time schedule



5.2 Task list

Task	Status
Project planning	Complete
3D design	Complete
Parts order	Complete
PCB design	Complete
Firmware	Complete
PDR write	Complete
New antennas design	Not started
Parachute design	In progress
Computer operation tests	In progress
Drop and flight tests	Not started
Gaining more sponsors and more popularity	In progress

5.3 Resource estimation

5.3.1 Budget

These are only estimated project costs:

Component	Price PLN (EUR)
First computer parts (PCB, ESP32, SD card module, Neo-6M GPS, HC-12, BME280, MPU6050, accessories)	120 zł (27 EUR)
ENS160 environmental sensor	100 zł (23 EUR)
Samsung INR1865-35E x1	25 zł (6 EUR)
Parachute (ripstop nylon, mounting string)	100 zł (23 EUR)
PLA+ Filament	75 zł (17 EUR)
Other 'private' resources (wires, solder, goldpins, etc.)	35 zł (8 EUR)
Total	455 zł (104 EUR)

5.3.2 External support

We got external support mainly from school teachers, like Mrs. Katarzyna Mazur – physics teacher. We also got some support from FSDM foundation and from our friends. The internet is also our great supporter.

6. Outreach program

First of all, we created our project website. It has a blog tab, where we post updates on the building process. We host it on Github Pages:
https://nevvman18.github.io/VarSat or https://tinyurl.com/varsat

After creating this site, we made a fundraising on an online service, available at:
https://zrzutka.pl/z/varsat1
It quickly became popular. Unfortunately, we found out that it may not provide sufficient money, so we wrote an email to FSDM Foundation and to our school Parents' Council. Both requests have been accepted.

Lastly, our coordinating teacher wrote a short text about our project and shared the link on the school's website and social media:

Photograph 22 - School Facebook Post

CanSat	
Dodał	Katarzyna Mazur
Data publikacji	2023-09-29
Treść	Szanowni Państwo, w tym roku szkolnym mamy swojego reprezentanta w międzynarodowym konkursie CanSat (zespół VarSat1, składający się z uczniów klasy 2c i 3c). CanSat to międzynarodowe wyzwanie Europejskiej Agencji Kosmicznej, podczas którego młodzież konstruuje symulatory sond kosmicznych oraz przeprowadza za ich pomocą badania naukowe. Uczniowie z naszej szkoły stoją przed wyzwaniem umieszczenia w CanSacie wszystkich podstawowych systemów składających się na satelitę – zasilania, modułu komunikacji oraz komputera pokładowego. Odpowiednio przygotowana sonda zostanie wyniesiona za pomocą rakiety. Dzięki projektowi CanSat uczestniczące Zespoły doświadczają wszystkich faz profesjonalnego projektu kosmicznego od wyboru celów misji, finansowania, promocji, zaprojektowanie CanSata, integracji komponentów, testowania systemu, raportowania, przygotowania do startu po analizę uzyskanych danych naukowych. W ramach pozyskania funduszy na niezbędne elementy, które zostaną umieszczone w sondzie nasz zespół VarSat1 zorganizował zrzutkę. Będziemy wdzięczni, jeśli Państwo zechcą wesprzeć nasz projekt choćby niewielką kwotą. Link do zrzutki: www.zrzutka.pl/z/varsat1 Zespół VarSat1 wraz z opiekunem.

Photograph 21 - Website Post

7. CanSat characteristics

Characteristics	Figure
Height of the CanSat	113 mm
Diameter of the CanSat	65 mm
Mass of the CanSat	301 g
Estimated descent rate	8 m/s
Radio transmitter model and frequency band	HC-12 | 434 MHz ISM band
Estimated time on battery (primary mission)	15 hours
Cost of the CanSat	405 PLN (92 EUR – exchange rate on 22.10.2023)