Space Applications Study Course
25.4.2023

MPA-NDE Semestral Report: Development of a Power PCB Control Board for CubeSatSim Educational Platform

Project Overview

At the beginning of the semester, a series of basic tests was conducted to familiarize students with the CubeSatSim software and the Raspberry Pi Zero platform. These tests included Automated Frequency Shift Keying (AFSK) Mode Tests, Slow Scan Television (SSTV) Tests, Frequency Shift Keying (FSK), and Binary Phase Shift Keying (BPSK) FoxTelem Tests. These experiments laid the foundation for understanding the communication and telemetry protocols commonly used in nanosatellite technology.

Based on the provided platform, the class identified an opportunity to improve the CubeSatSim design by proposing a new power control board PCB. The aim was to enhance the power management capabilities of the simulated CubeSat, mainly to offer alternative options to provide power to the whole system. Throughout the remainder of the semester, the focus was centered around the design and development of this new power control board. The project was divided into several phases: initial testing, research, feasibility study, prototyping, schematic design, software development, PCB design, and fabrication & testing. By the end of the semester, the class successfully completed the design of the PCB and partially demonstrated its functionality.

Phase Zero: Initial Testing

In its initial state, the CubeSat consisted of a Raspberry pi zero and a CubeSatSim Lite board. These components were placed in the basic frame structure. As can be seen on Figure 1.

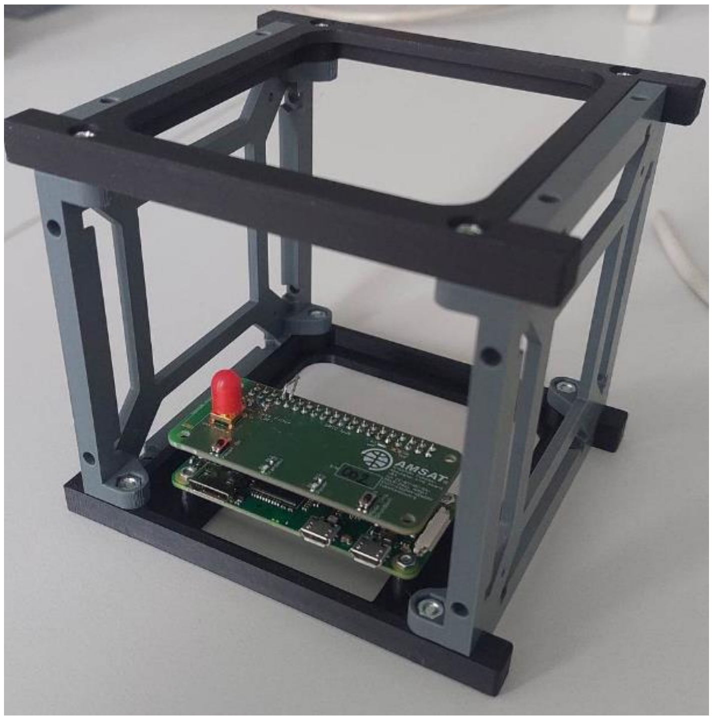


Figure 1 Initial state

The communication tests were performed first with the SDRsharp program. An RTL-SDR R860T2 USB receiver was connected to the computer, with the help of which this communication was successfully completed. Testing of sending basic telemetry data was done using the FoxTelem program. A Finally, a test of sending a picture using the MMSSTV program was also carried out. This program works in cooperation with SDRsharp, with which it was connected using a virtual audio cable and decoded the image from the audio signal. All tests were successful and created a basis for further improvement possibilities

Phase One: Research

As CubeSats are very small, as they aim to be very cost-effective, one of the key challenges in their design is ensuring reliable and efficient ways to generate power and coming up with methods for power management.

Solar Panels are the most common power source for CubeSats. There are several types of solar panels, including monocrystalline, polycrystalline, and thin-film panels. Typically, CubeSats use monocrystalline (often multi-junction) layers as they offer the highest efficiency. An exemplary structure is a triple-junction GaAs cell. These solar cells are lightweight, have a high power-to-weight ratio, and can provide power for the duration of the CubeSat mission. Depending on the CubeSat's power requirements, solar panels can be mounted on one or more sides of the satellite or even be deployable to maximize energy generation.

Radioisotope Thermoelectric Generators (RTGs) are an alternative power source for CubeSats, particularly for deep-space missions due to a low-starlight environment. They generate electricity by converting the heat released from the natural decay of radioisotopes, such as plutonium-238. Power is converted via thermocouples. Although RTGs offer a long-lasting and stable power source, their use in CubeSats is limited due to their high cost, regulatory restrictions, and potential safety concerns.

Other emerging energy harvesting methods involve **energy harvesting from alternative sources.** They aim to capture and convert ambient energy from the satellite's environment into usable electrical power. Potential energy harvesting sources for CubeSats include vibrational energy from the satellite's deployment mechanism or other onboard systems, thermal gradients from satellite components, and even electromagnetic energy from radiofrequency signals. While energy harvesting technologies are still in the early stages of development, they hold promise for supplementing traditional power sources and increasing CubeSat mission lifetimes.

Batteries serve as energy storage devices for CubeSats, ensuring that power is available during periods of eclipse or when solar energy generation is insufficient. Commonly used batteries for CubeSats include lithium-ion (Li-ion), lithium-polymer (LiPo), and lithium iron phosphate (LiFePO4) batteries. These battery types offer a high energy density, long cycle life, and a relatively stable voltage output.

Phase Two: Feasibility Study

Solar energy is an abundant and renewable resource, making it an ideal power source for CubeSats. As while they operate in Low Earth Orbit (LEO), they are exposed to sunlight for a significant portion of their orbital period, which can be efficiently converted into electrical energy, providing a continuous supply of power to the satellite. Triple-junction GaAs cells have a well-established track record in space applications, as they offer excellent power-to-weight ratios. However, considering the educational nature and budget constraints of the project, triple-junction cells can be beneficially replaced with their low-cost, less-efficient alternatives.

Batteries serve as an essential complement to solar panels, providing energy storage capabilities that ensure power is available during periods when solar energy generation is insufficient or completely absent. This energy storage solution ensures the continuous operation of the CubeSat's systems and instruments, allowing the system to operate without interruption.

Solar panels and batteries are relatively simple for integration, making them suitable for our educational test platform, where ease of use and accessibility are important factors. Students can focus on learning the principles of power management and satellite design without being overly precautious due to risks associated with such expensive components.

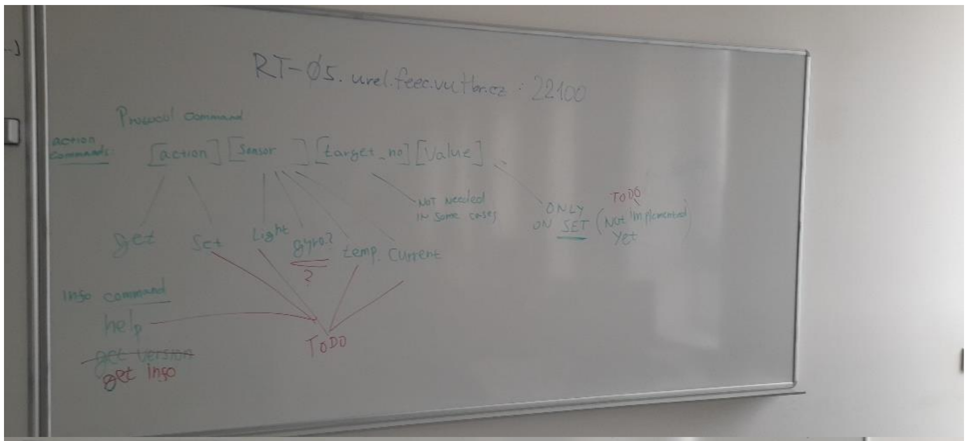
Phase Three: Prototyping

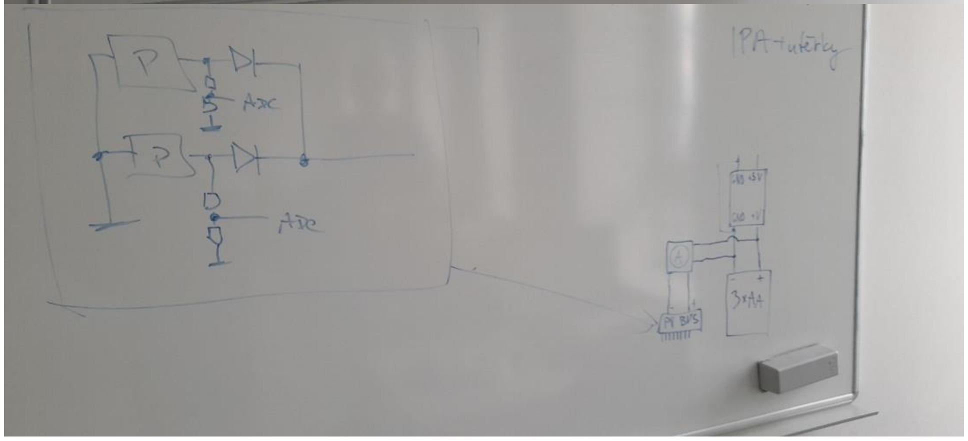
The main body of the prototyping work consisted of verifying the feasibility of the planned circuits, as well as the verification of proper internal and external communication of the satellite.

The first phase of prototyping included the creation of several simple battery management circuits:

- Case 1: 2 AA Batteries without a DC-DC converter
- Case 2: 3 AA Batteries without a DC-DC converter
- Case 3: 2 AA Batteries with a DC-DC converter
- Case 4: 3 AA Batteries with a DC-DC converter

During this phase it was shown that the satellite raspberry pi zero module cannot be powered purely from battery power, as it causes charging and discharging of the raspberry capacitors, causing looping restarts. Therefore, a DC-DC converter was used to convert from the typical 2.7 V for 2xAA and 4.1V for 3xAA to a stable source of 5V.





It should be noted that during the creation of this circuit a new safety requirement was created, since the DC-DC circuit immediately gets destroyed if the input voltage is higher than the set output voltage. Therefore, all further testing requires that the conversion rate is known and within the requirement.

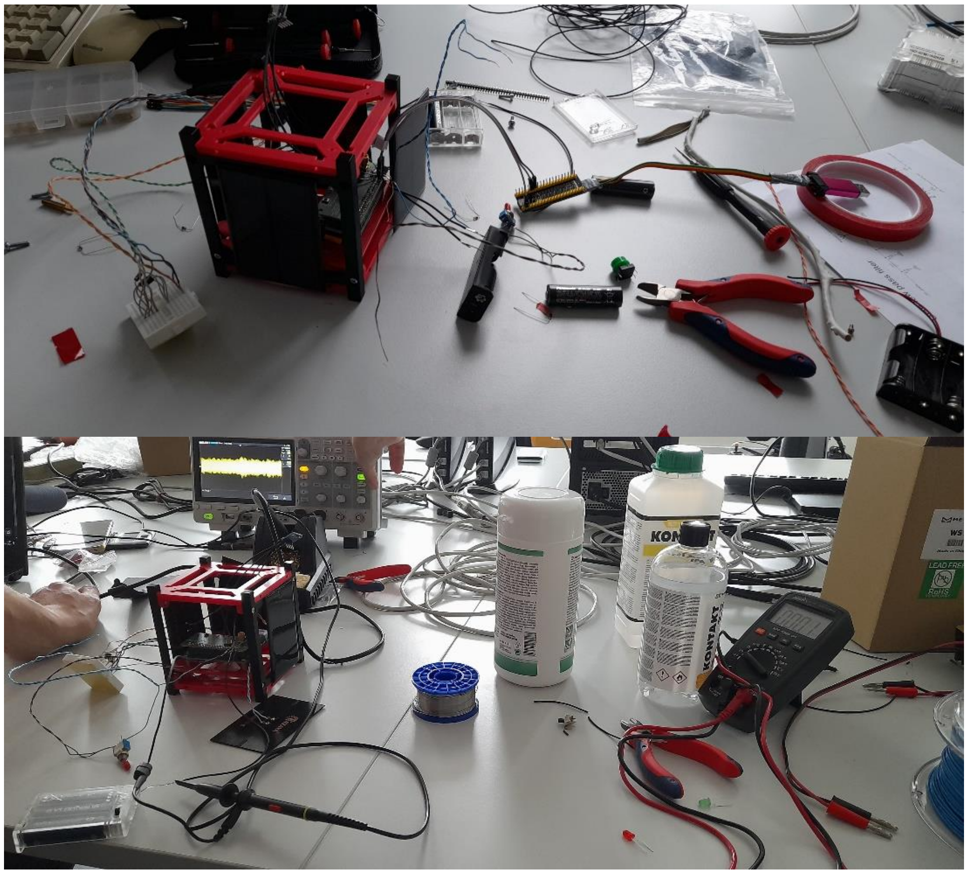
Once the basic circuit had been created, the solar cells were prepared for use and the battery management system had been extended.

- Case 5: 2 AA Batteries with a DC-DC converter Solar Panel circuit (normal diodes)
- Case 6: 2 AA Batteries with a DC-DC converter Solar Panel circuit (schottky diodes)
- Case 7: 2 AA Batteries without a DC-DC converter Solar Panel circuit (schottky diodes)

The solar panel circuit consisted of the following components:

- 4x Solar Panels 90x60mm
- 4x Diodes for circuit protection (normal, later schottky)
- On/off switch for controlling current flow from PV circuit to battery circuit

The final form which was successfully tested and verfied was Case 6, which was also used in the integrated BMS/PSU circuit.



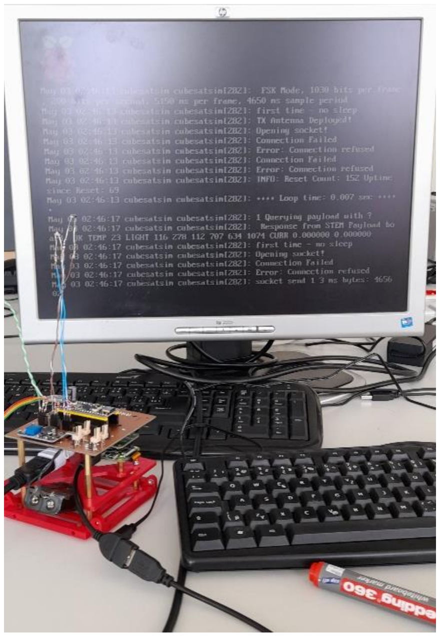
During the prototyping phase of the circuit, the following outputs were gathered:

- The batteries are insufficient to power the raspberry pi by themselves
- If the voltage of the batteries falls below 2.4V the system experiences brownout
- The maximum power output of the solar paneling of the satellite is in units of mA, possibly showing insufficient power generation.
- Schottky diodes slightly increase the power output of the PV panels.

The third phase of the prototyping work consisted of ensuring communication occurred between the PSU blackpill module and the main raspberry pi module.

In the initial phases of prototyping, UART protocol was chosen instead of i2c for several reasons, including it being considered more stable for spacecraft, however the main reason being the number of channels available if the blackpill module had been chosen.

There were two parts of this effort, the first part consisted of the circuit team creating a communication protocol that would be transmitted to the raspberry pi. The second part consisted of the prototyping team modifying the raspberry pi code to accept and correctly interpret the received data.



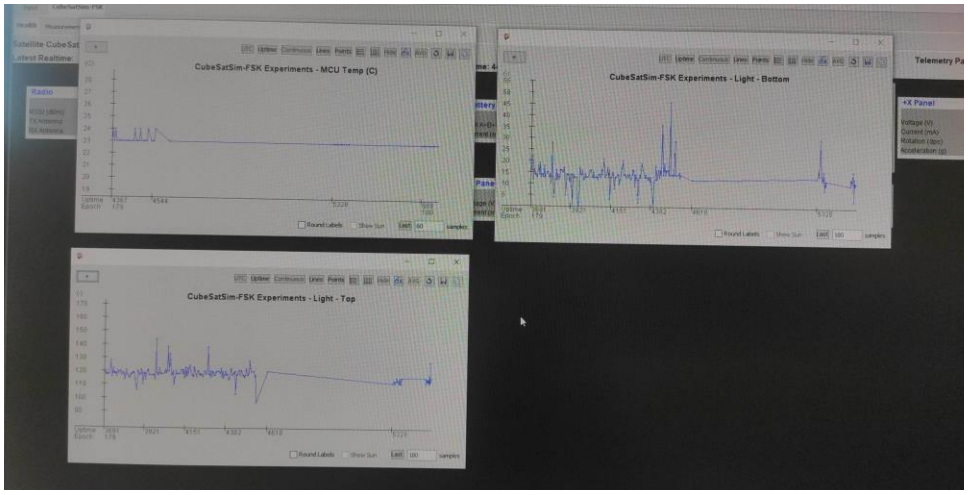
```
May 03 02:46:13 cubesatsim cubesatsim[282]:  FSK Mode, 1030 bits per frame
, 200 bits per second, 5150 ms per frame, 4650 ms sample period
May 03 02:46:13 cubesatsim cubesatsim[282]: first time - no sleep
May 03 02:46:13 cubesatsim cubesatsim[282]: TX Antenna Deployed!
May 03 02:46:13 cubesatsim cubesatsim[282]: Opening socket!
May 03 02:46:13 cubesatsim cubesatsim[282]: Connection Failed
May 03 02:46:13 cubesatsim cubesatsim[282]: Error: Connection refused
May 03 02:46:13 cubesatsim cubesatsim[282]: Connection Failed
May 03 02:46:13 cubesatsim cubesatsim[282]: Error: Connection refused
May 03 02:46:13 cubesatsim cubesatsim[282]: INFO: Reset Count: 152 Uptime
since Reset: 69
May 03 02:46:13 cubesatsim cubesatsim[282]: **** Loop time: 0.007 sec ****
*
May 03 02:46:17 cubesatsim cubesatsim[282]: 1 Querying payload with ?
May 03 02:46:17 cubesatsim cubesatsim[282]:  Response from STEM Payload bo
ard: OK TEMP 23 LIGHT 116 278 112 707 634 1074 CURR 0.000000 0.000000
May 03 02:46:17 cubesatsim cubesatsim[282]: first time - no sleep
May 03 02:46:17 cubesatsim cubesatsim[282]: Opening socket!
May 03 02:46:17 cubesatsim cubesatsim[282]: Connection Failed
May 03 02:46:17 cubesatsim cubesatsim[282]: Error: Connection refused
May 03 02:46:17 cubesatsim cubesatsim[282]: socket send 1 3 ms bytes: 4656
02
```

After several attempts a connection had successfully been made and the correct telemetry data has been confirmed to have arrived at the testing ground station. This phase took the most amount of time as several issues had arisen:

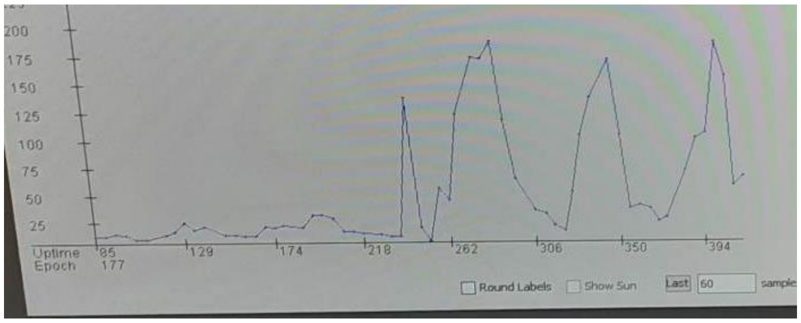
- The UART protocol had to precisely replicate the behavior of the expected STEM payload.
- Blackpill resets have shown to be transmitting garbage data to the raspberry.
- Raspberry seems to partially block RX/TX communication in certain cases, such as unexpected characters at the ttyAMA0 terminal.
- The parts of the code containing the communication with the STEM payload had to be found and reverse engineered.

The communication protocol had been determined to work as follows:

1. CubeSatSim sends out aarestart signal to the payload (a single 'R') to tty/AMA0 at 115200 bauds
2. Payload restarts (dummy restart) and responds 'OK'
3. If successful, CubeSatSim will periodically (approx. Every 5s) send out a query ('?') signal to the payload
4. Payload responds with 'OK TEMP <Val1> LIGHT <Val1> <Val2> <Val3> <Val4> <Val5> <Val6> CURR <Val1> <Val2>'
5. CubeSatSim processes the data according to csv files in spacecraft/FoxTelem<version> folders, and transmits them
6. FoxTelem on the ground station receives this data


CubeSatSim-FSK Experiments - MCU Temp (C)
CubeSatSim-FSK Experiments - Light - Bottom
CubeSatSim-FSK Experiments - Light - Top

Successful light sensor reading during constant cubesat rotation:



Phase Four: Schematic Design

In this phase, the integration of solar panels and batteries to power the Raspberry Pi Zero was laid out. The schematic design provides a foundation for understanding the connections and components involved in creating a base energy source for the Raspberry Pi. The design aims to provide the test platform with power generation and power storage while maintaining simplicity and cost-effectiveness. The final schematic is displayed in Figure 1.

Schottky diodes act as blocking diodes, preventing the reverse flow of current from the controller or other parts of the circuit back to the solar panels. This ensures that the energy stored in the batteries isn't drained back into the solar panels. Additionally, these diodes have a relatively low forward voltage drop (typically 0.2-0.5V), which minimizes power loss across the diode when conducting current. This is beneficial for the CubeSat, as power efficiency is crucial due to the limited energy available from the solar panels.

A **polyfuse** is included in series with the battery to provide protection against overcurrent and short-circuit events. It is a crucial safety component that can help prevent damage to the battery and other electronic components in the circuit. When the current in the circuit exceeds a predefined threshold, the polyfuse's resistance increases rapidly, effectively limiting the current flow.

A **DC-DC converter** MT6308 is used in the circuit for the purpose of regulating and converting the voltage level. The microcontroller requires a different voltage level than that provided by the solar panels and the battery. The converter steps up the input voltage to 5V to match the required voltage level for the microcontroller.

A **P-MOS Transistor** allows controlled power delivery to the Raspberry Pi Zero. By controlling the gate of the PMOS transistor, the other MCU can enable or disable power to the Raspberry Pi Zero based on specific conditions or requirements.

The **MCU** can monitor the current and voltage levels in the system and disable the PMOS transistor if an overcurrent or overvoltage condition is detected.

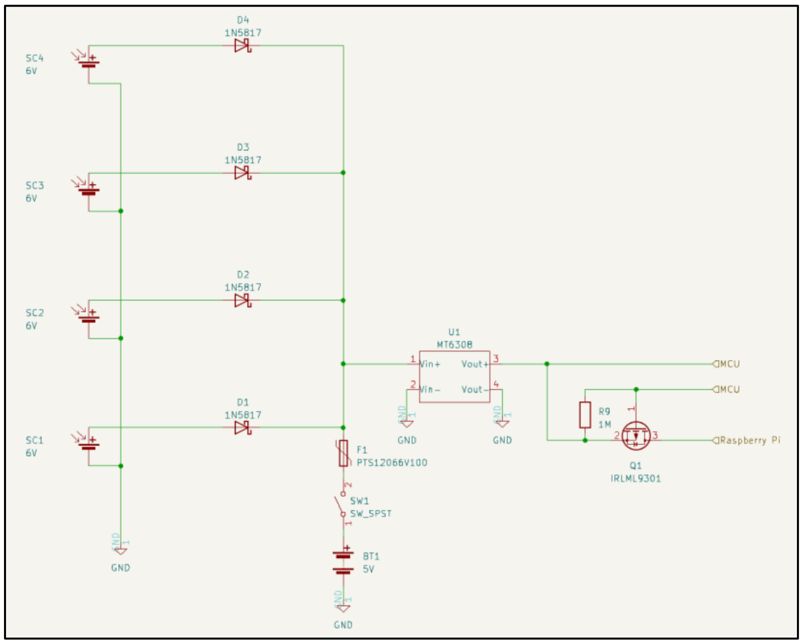


Figure 2. - Schematic design of the power feed.

Phase Five: Software Development

Based on the prototype and the project requirements we designed a code. The code is written in LibOpenCM3, which is a library focused on development of Cortex-M3 architecture devices. The code consists of main part which handles the communication. Based on received data the code executes one of following operations:

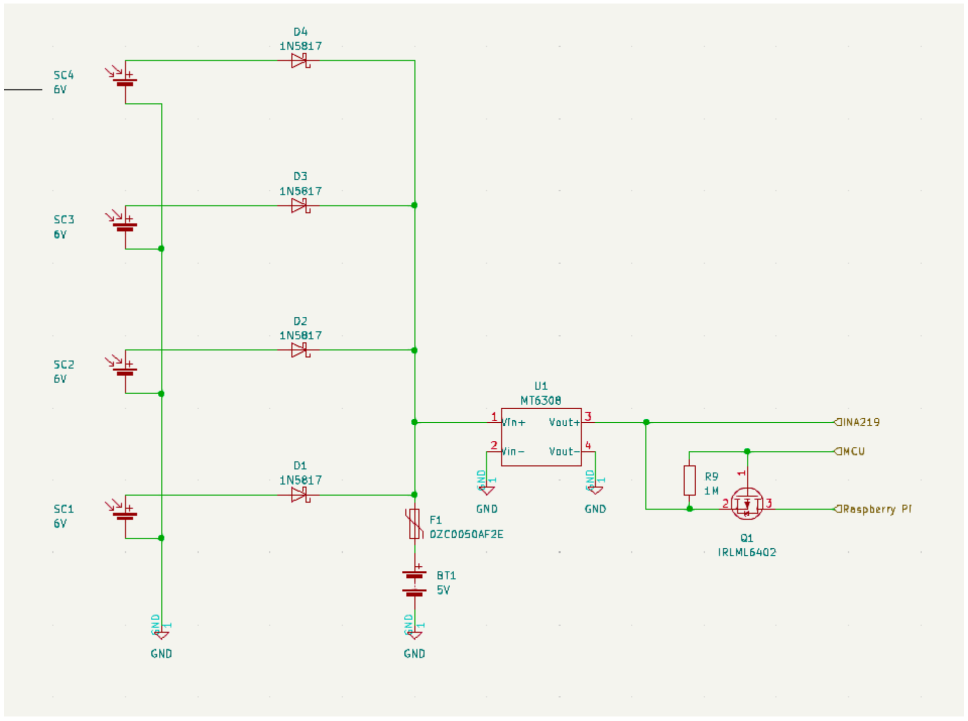
- ‘R’: Device responds with ‘OK’ and then dummy resets itself
- ‘?’: Device returns a string containing telemetrics
- ‘help’ + enter: Debug command - device returns a list of manual commands (every of the command needs to be sent manually by enter)

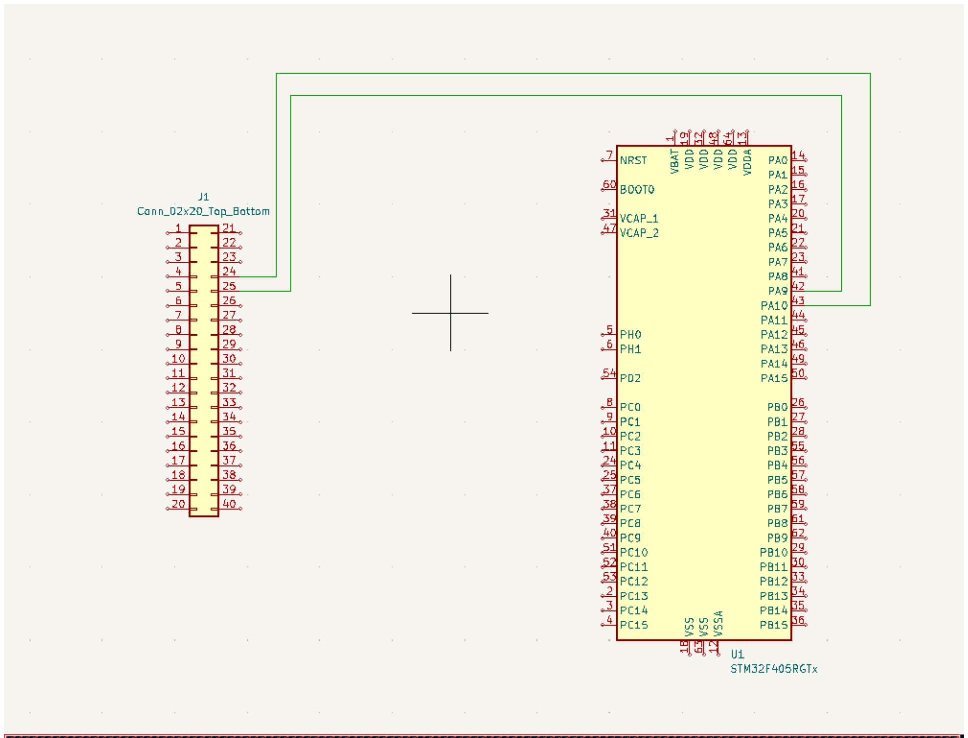
The code is designed in a way that ‘R’ and ‘?’ request has the most priority and is handled by IRQ. Every other command is also handled by IRQ but is not executed until enabled (debug commands are commented by default) and the character for ‘enter’ appears.

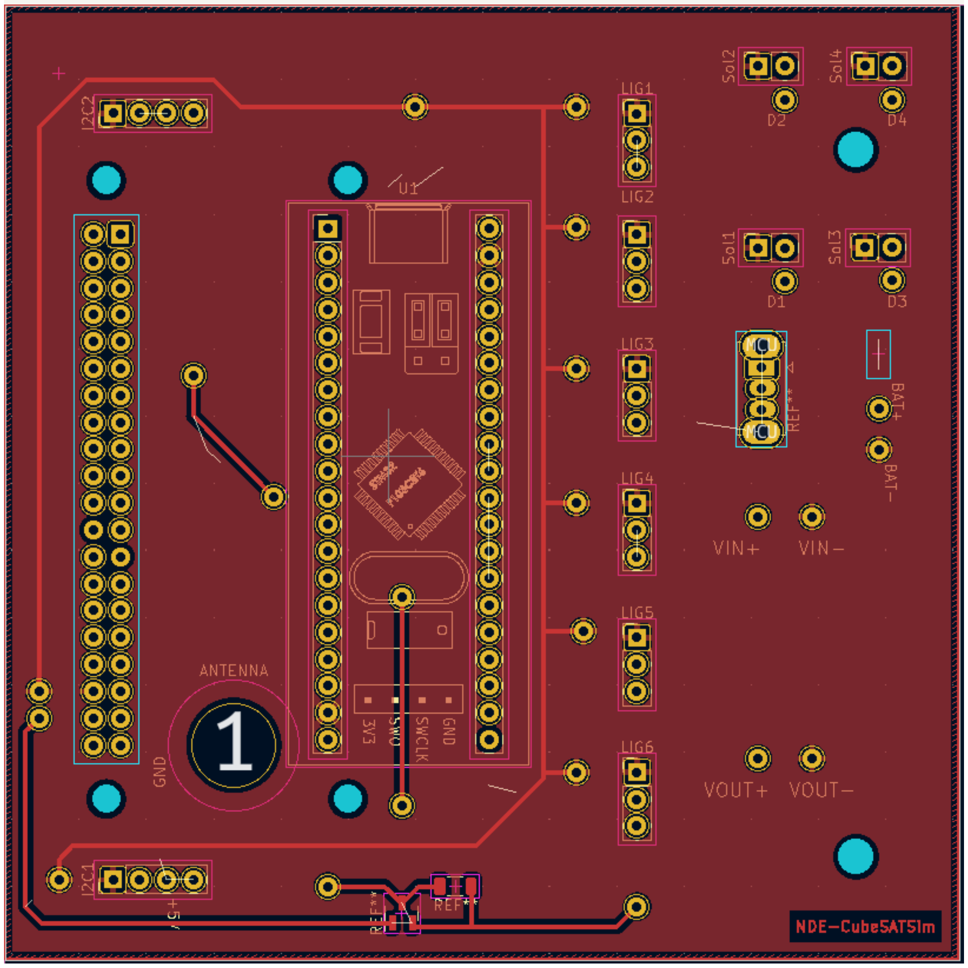
The code is designed around IRQ, which handles most of the program's workflow. Other necessary things like peripheral setup are done usually only once.

Phase Six: PCB Design

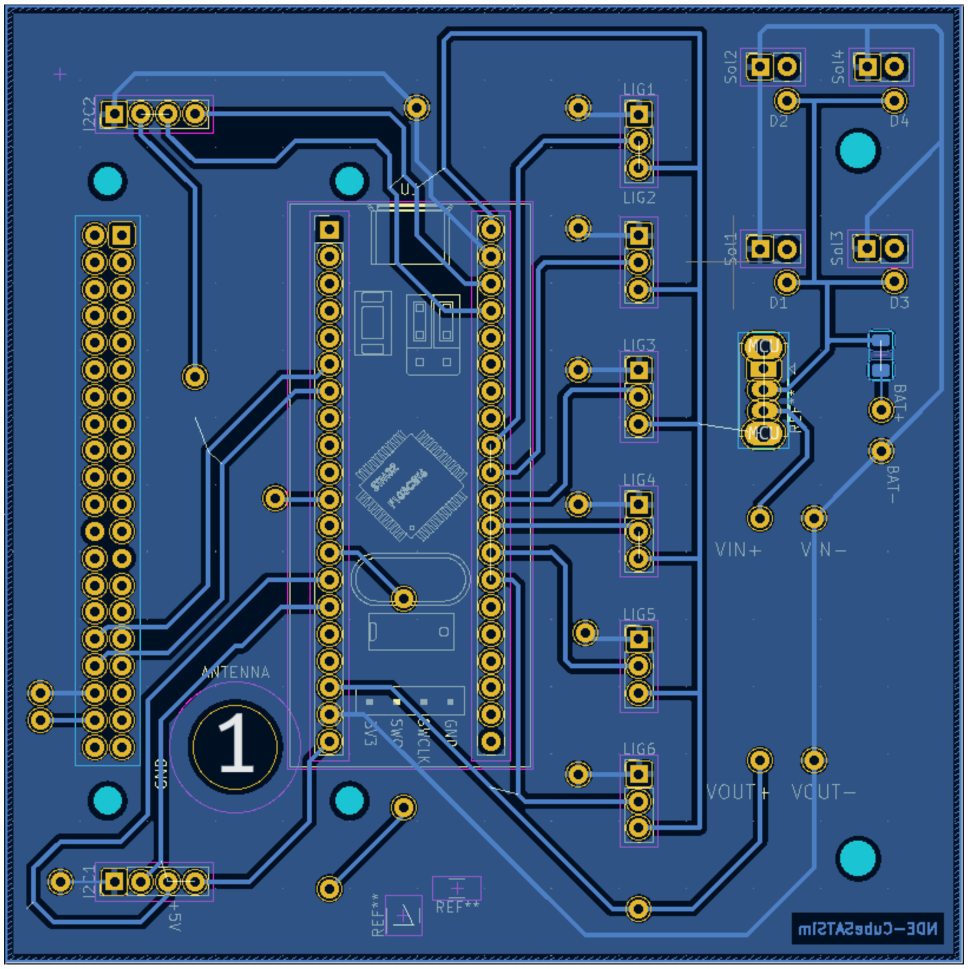
Same as the code – based on the prototype and the project requirements, we designed and created a printed circuit board. The design itself was divided into 2 parts – circuit design and PCB design in which the circuit board was divided between the BMS (battery management system) and the peripherals connection design. The most visible part of the PCB is the hole placed between the MCU and RPi connector. This hole is designed to let the antenna through.







D. Sedláček (221347), A. Bekeč (221096), M. Borovička (208436), M. Picka (201573), M. Roman (222951), J. Veverka (205866)



Phase Seven: Fabrication & Testing

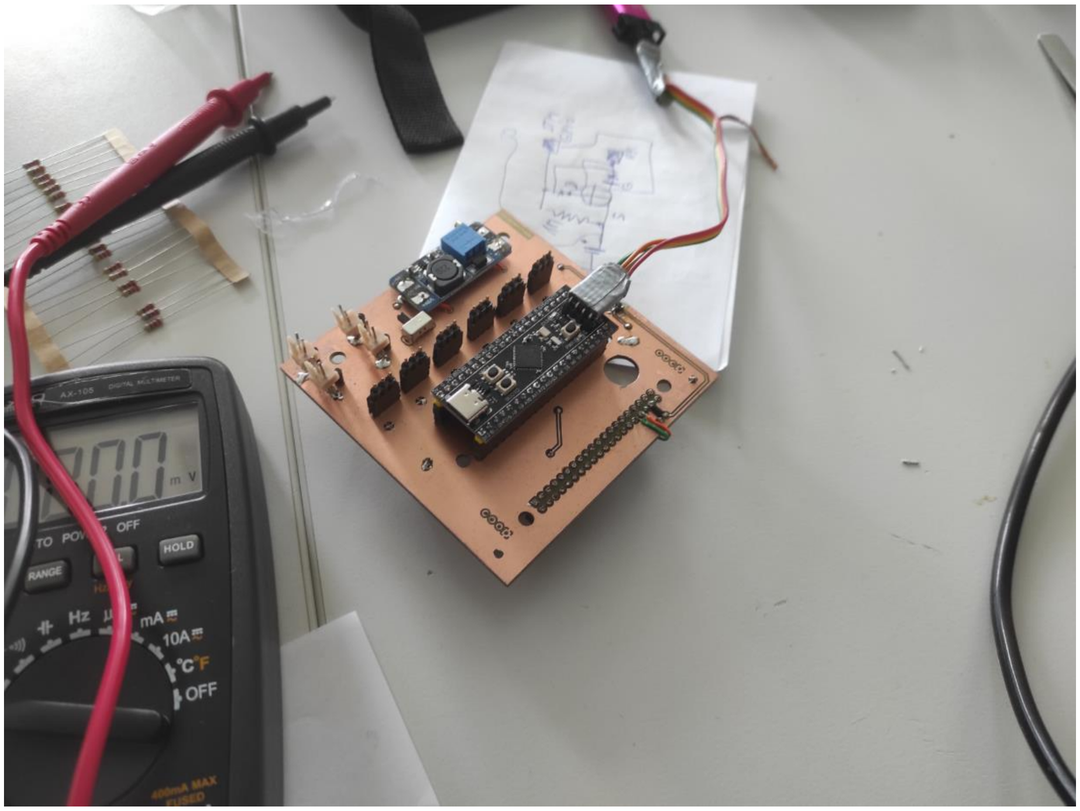
During the testing phase we encountered multiple problems.
We successfully managed to fix everything and got the system running.

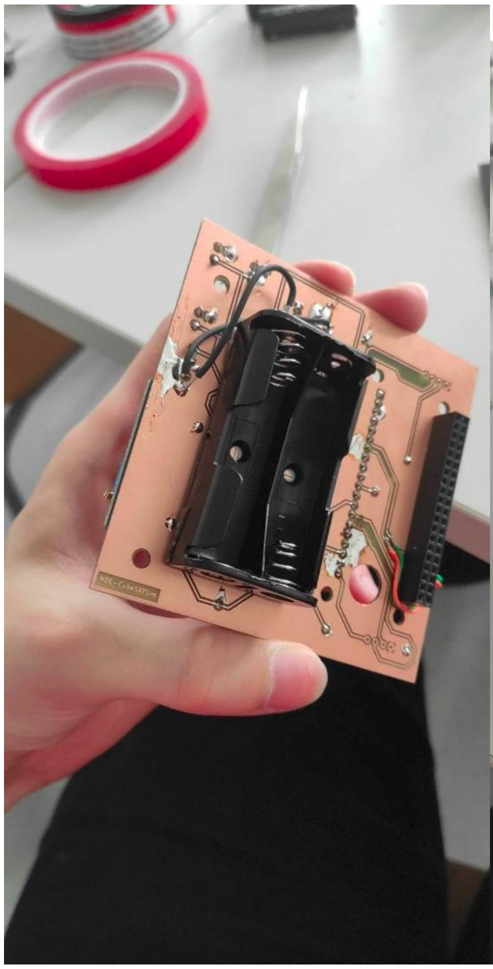
List of encountered problems:

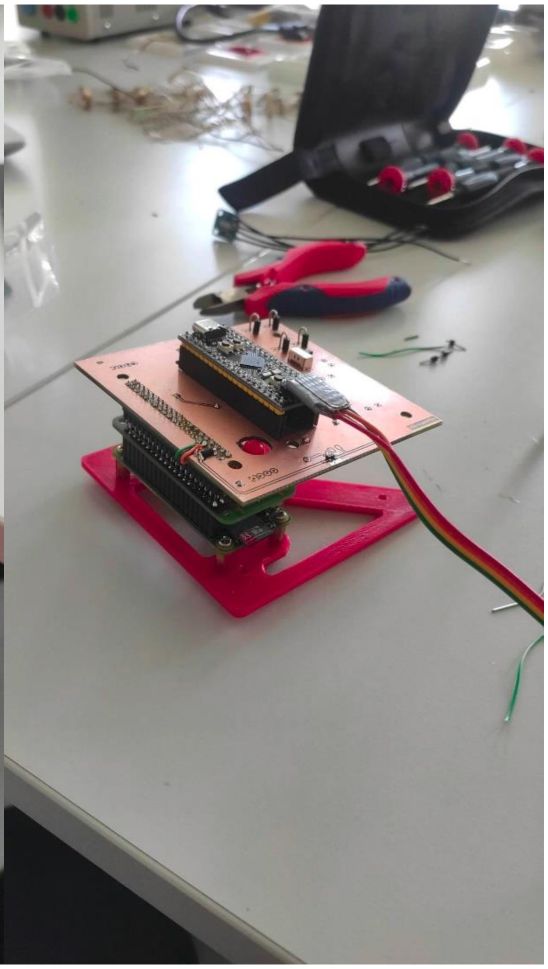
- <u>PCB Design</u>:
  - Wrongly designed connectors for solar panels – resolved by soldering the connector onto the diode and piece of wire.
  - Connector for RPi header designed for wrong side – fixed by wires bridging the gap on one side to another.
  - MOSFET orientation design was not according to the datasheet – fixed by rotating the MOSFET.
  - Power system problems – Fixed by soldering a parallel capacitor to the powerline and removing the fuse, resulting in stable power supply.

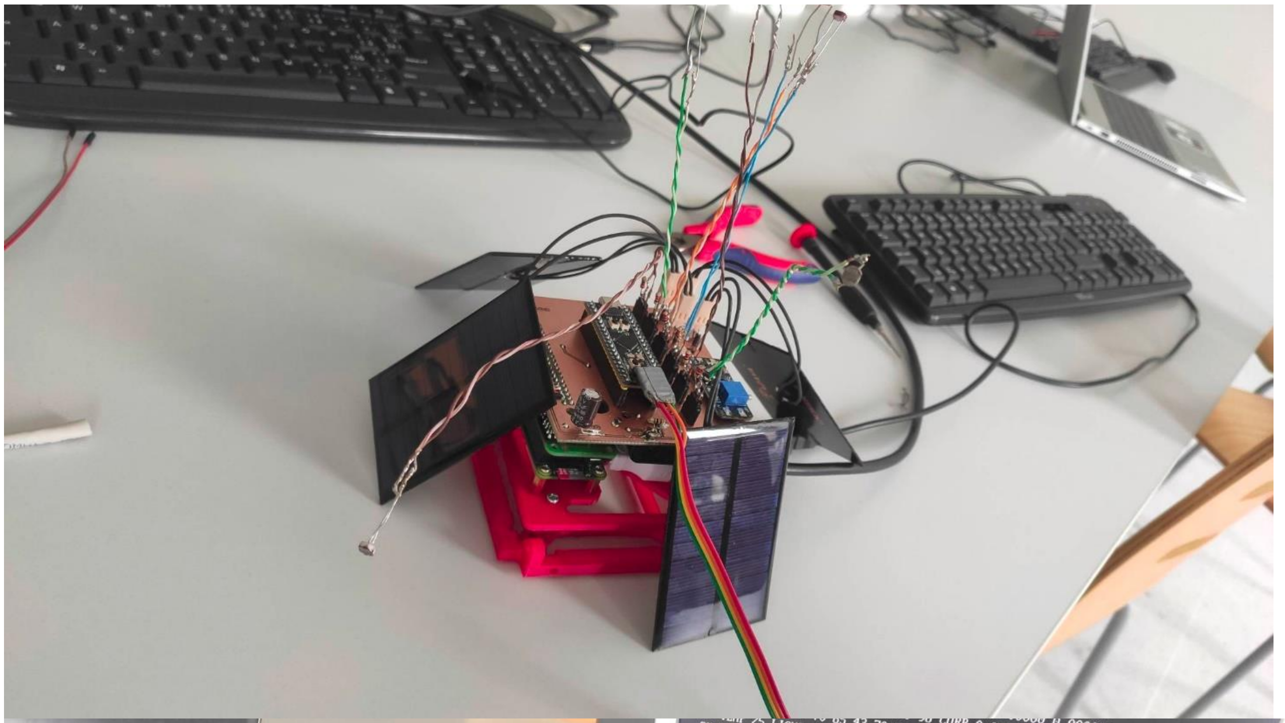
  - Redesign of RPi header placement needed – to enable easy access to HDMI and microUSB ports.
- <u>SW + FW</u>:
  - Communications problem (stopped responding after a while) – resolved by implementing priority line in ISR.
  - Communications problem (payload content incomplete) - resolved by increasing the size of the buffer containing the message.
  - Communications problem (echo) - resolved by disabling the TX response.
  - Communications problem (waiting for 'enter') - On commands 'R' and '?' the MCU was waiting for the 'enter' character. Since these commands are sent by computer automatically and don't wait for confirmation, the problem was resolved by handling in poll or in ISR.
  - RPi problem (wrong payload name) - renamed.
  - RPI problem (service fails to start) - resolved by reinstalling the CubeSatSim files.
  - RPi declining communication from MCU – resolved by changing the batteries.
  - RPi displaying (almost) static data – caused by buffer overflow which happened because of too frequent data sending from the MCU. Resolved by slowing the data flow.

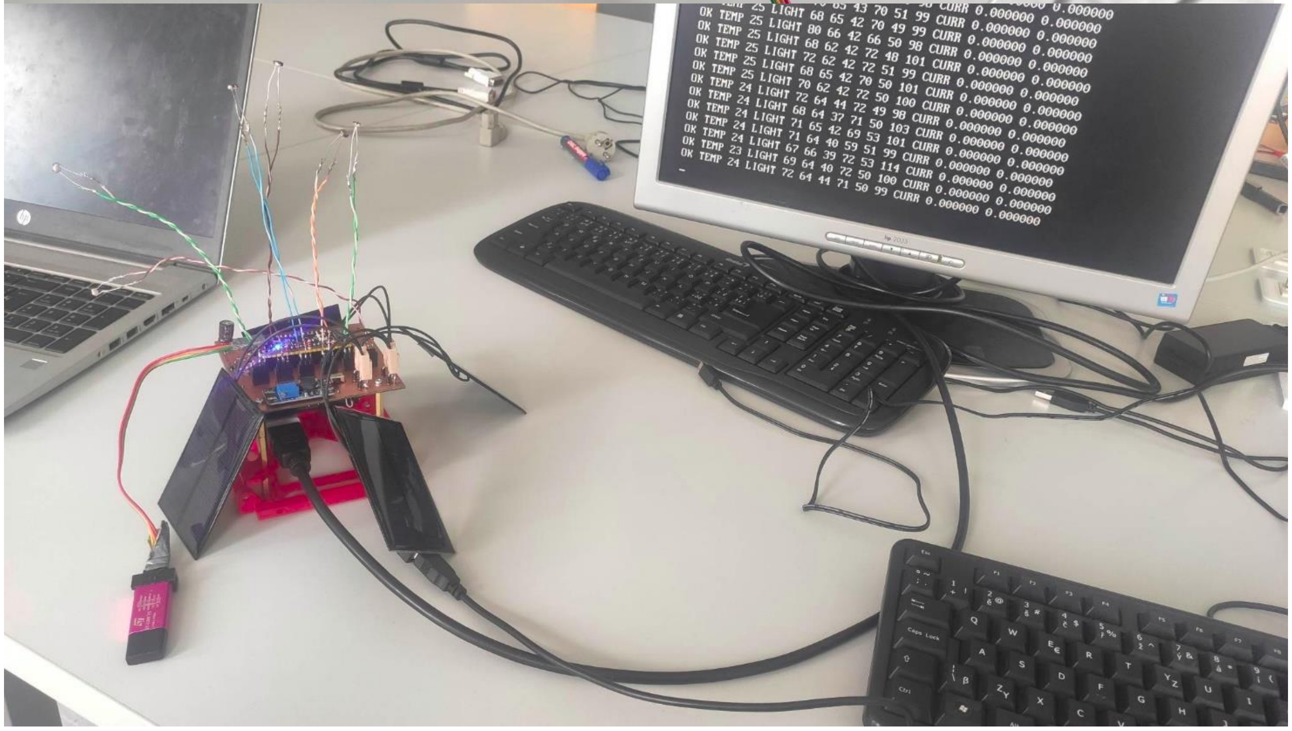
rosetta
LIVING
WITH A COMET







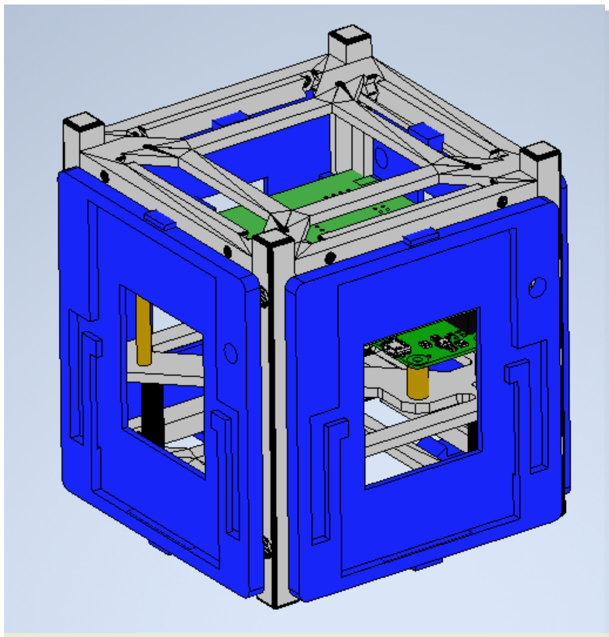


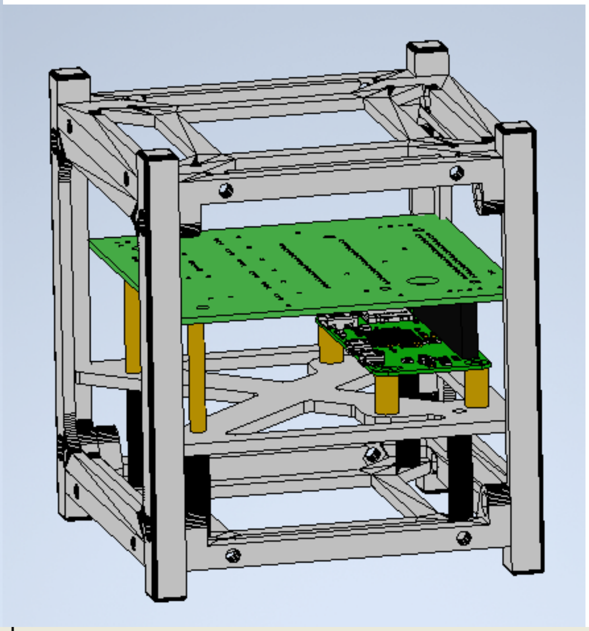


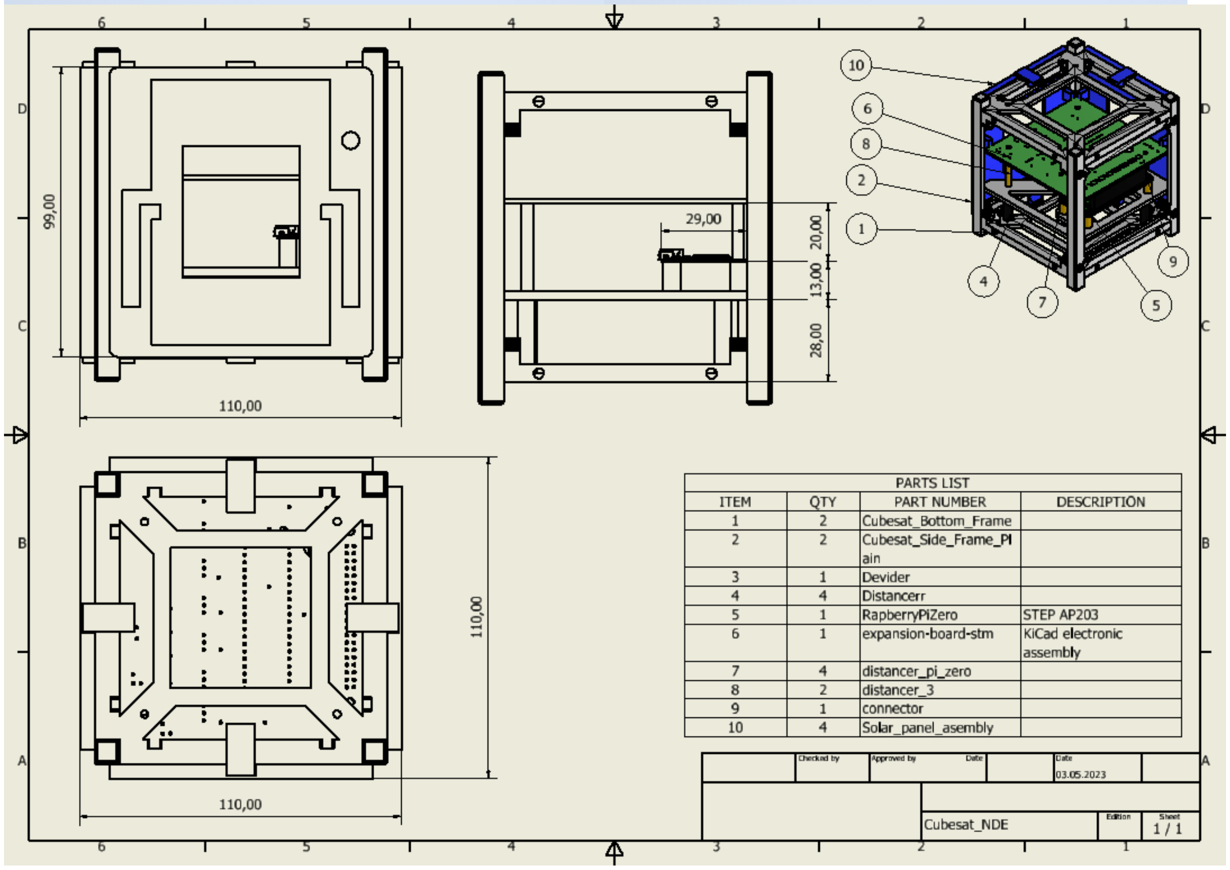
```
     25 LIGHT [illegible] 65 43 70 51 99 CURR 0.000000 0.000000
OK TEMP 25 LIGHT 68 65 42 70 49 99 CURR 0.000000 0.000000
OK TEMP 25 LIGHT 80 66 42 66 50 98 CURR 0.000000 0.000000
OK TEMP 25 LIGHT 68 62 42 72 48 101 CURR 0.000000 0.000000
OK TEMP 25 LIGHT 72 62 42 72 51 99 CURR 0.000000 0.000000
OK TEMP 25 LIGHT 68 65 42 70 50 101 CURR 0.000000 0.000000
OK TEMP 25 LIGHT 70 62 42 72 50 100 CURR 0.000000 0.000000
OK TEMP 24 LIGHT 72 64 44 72 49 98 CURR 0.000000 0.000000
OK TEMP 24 LIGHT 68 64 37 71 50 103 CURR 0.000000 0.000000
OK TEMP 24 LIGHT 71 65 42 69 53 101 CURR 0.000000 0.000000
OK TEMP 24 LIGHT 71 64 40 59 51 99 CURR 0.000000 0.000000
OK TEMP 24 LIGHT 67 66 39 72 53 114 CURR 0.000000 0.000000
OK TEMP 23 LIGHT 69 64 40 72 50 100 CURR 0.000000 0.000000
OK TEMP 24 LIGHT 72 64 44 71 50 99 CURR 0.000000 0.000000
_
```

Structure and design

CubeSat structure was designed from 3d printed parts with modularity in mind. Inside CubeSat houses modular and height adjusted platforms for attachment of components. Outside of the CubeSat has 4 modular solar panels and resistor holders, allowing quick and easy disassembly.

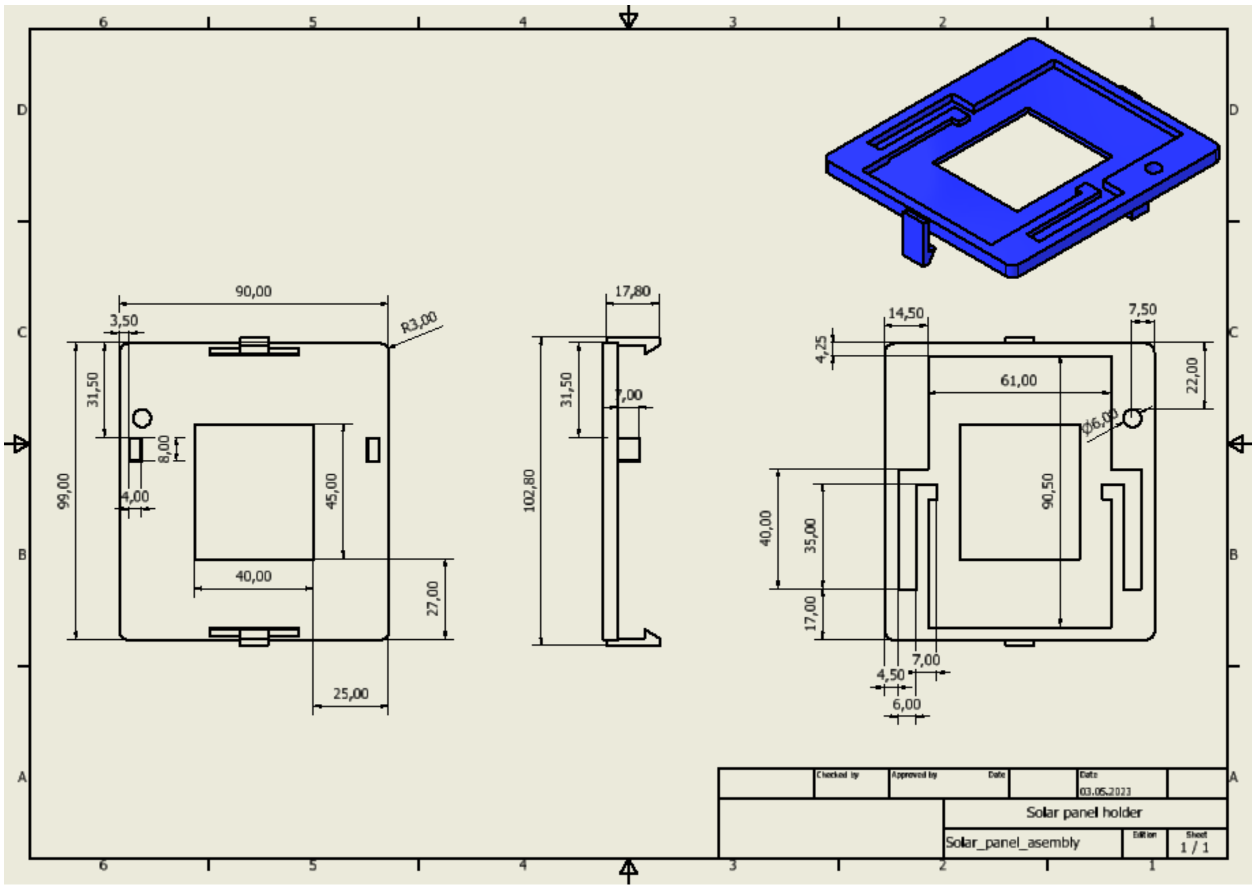






PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	2	Cubesat_Bottom_Frame	
2	2	Cubesat_Side_Frame_Plain	
3	1	Devider	
4	4	Distancerr	
5	1	RapberryPiZero	STEP AP203
6	1	expansion-board-stm	KiCad electronic assembly
7	4	distancer_pi_zero	
8	2	distancer_3	
9	1	connector	
10	4	Solar_panel_asembly	

Side-solar panels housing:



Final assembly of CubeSat demonstrator