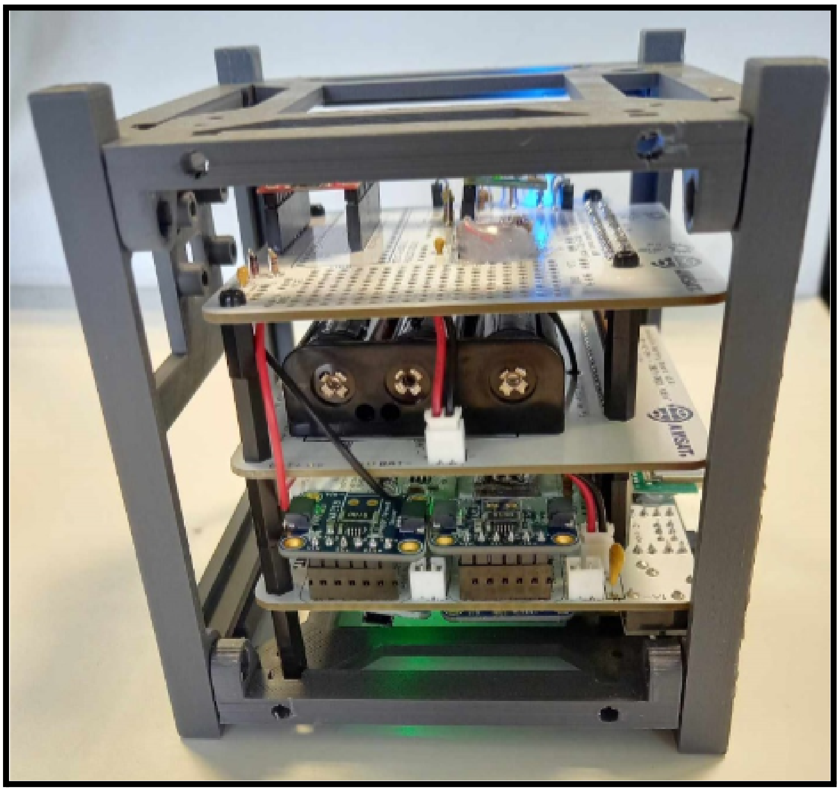
Report on CubeSat Construction and Utilization Issues and Solutions.
Nimbus Research Center.

Prepared by: HUGUET Yann
Date: 21/11/2023



Abstract:
This technical report documents the challenges faced and the solutions devised during the construction and operational phases of the CubeSat project. The report highlights critical issues encountered, their root causes, and the implemented solutions, aiming to provide insights into the development process and improvements made to enhance CubeSat functionality.

Table of contents:

1. Introduction.

We will exclusively discuss the problems encountered and the solutions found in our project, which is divided into two distinct stages. The initial stage involved constructing a CubeSat using information provided for the AMSAT CubeSatSim prototype accessible at [CubeSatSim repository](). We meticulously followed the provided steps and successfully replicated the foundational structure.

The subsequent stage revolved around addressing various issues and implementing solutions. Two primary problems were identified. Firstly, the challenge of achieving real-time visualization of our CubeSat in space through the use of a gyroscope. This 3D visualization was intended to be accessible on a website and update dynamically as the ground station received information.

Secondly, our focus was on prototyping a CubeSat capable of orienting itself in space using reaction wheel principles. This endeavor required substantial restructuring of the CubeSat to accommodate two motors and integrate new electronic circuit components.

---

2. CubeSat Construction Challenges.

- Difference in components.

During the construction of the CubeSat, we encountered various issues, with the first being the discrepancy between the components on the PCB we manufactured based on the data from https://github.com/alanbjohnston/CubeSatSim.git and the more recent components we had. For instance, notable differences were found in the "USB Boost 5v" and the "Micro USB breakout" components.

To resolve this type of incompatibility, we adopted a straightforward solution. We connected the corresponding parts of the components to the correct locations on the board using straps (wires). An additional technique to ensure stability and prevent excessive movement or potential for false contacts was to apply hot glue underneath. This helped secure the straps firmly and prevent any loose connections, ensuring maximum isolation.

This method allowed us to mitigate the mismatch between components effectively, ensuring proper functionality despite the disparities between the expected and the available parts.

- Surface-mounted part.

During the CubeSat construction, there is also a section involving surface-mounted components. It's crucial to exercise extreme care and precision during soldering. Improper soldering might result in functional issues despite seeming to work, as even minor misalignments can cause significant problems (as the filter won't be precise at all). To ensure proper functionality, use continuity tests and meticulously verify component values to ensure none have been damaged by excessive heat during soldering.

- Reversal of cable polarity

During the CubeSat construction, we observed that our cables linked to the JST connectors were in opposite positions in terms of colors when perfectly following the PCB layout. This presented an issue for visualization. Consequently, we straightforwardly flipped all the connectors so that the power was correctly associated with red and the ground with black.

---

3. Ground station issue.

- Image.

It is crucial to exercise caution when uploading the image into the ground station and strictly use the image named as follows: "Fox-in-box-v3.iso.gz," available as per the tutorial instructions. Additionally, if you are using Balena Etcher, encountering an error message at the end is completely normal. This does not necessarily indicate a failed image upload.

Attention, updating the image on the Raspberry Pi of the ground station, following the tutorial, will result in the absence of the Visualisation.service and the Rasp.py Python script, essential components for the visualization project. In such a scenario, it is necessary to reconfigure these elements by carefully following the instructions outlined in the readme file provided within this directory. This process will help restore the required services and scripts for the visualization project to function properly.

- Compatibility.

Caution must be exercised regarding the specific model of Raspberry Pi used for the ground station. In our experience, we temporarily switched from using a Pi 4 Model B to a Pi 3. Initially, everything appeared to function properly concerning the software on various interfaces. However, when utilizing Foxtelem, the data reading tool, we encountered an issue where receiving data became impossible. This turned out to be a compatibility problem specifically related to the Pi model change.

---

4. Soft on Ground Station issue.

- FoxTelem.

The primary software we utilize, Foxtelem, facilitates signal reception visualization. By switching to mode 2 using the button, we transmit all necessary data. These data points update alongside the "payload" count, gradually increasing over time. However, there are instances where the count stops increasing while everything else appears to function normally. In such cases, observing the eye diagram is crucial. If the diagram isn't stable despite apparent functionality, the reception ceases. The only current solution is to reboot the CubeSat (turning it off and then on) to reinitiate communication. As of now, this is the sole method to resolve the issue, as we have yet to identify the root cause.

- Battery.

There have been instances where the ground station received the signal visible in red on Foxtelem, but visualizing the reception of data (in blue), which typically generates waves representing the transition between 0 and 1, was impossible. In essence, we solely observed the red spike, indicating communication but no data passage or payload incrementation. Initially, we suspected the ground station as the source of the issue. However, after investigation, the problem originated precisely from the CubeSat, specifically from its battery. This occurrence arises because the battery has sufficient power to establish communication but lacks the capacity to transmit data. The resolution involves simply replacing or recharging the batteries.

- Payload.

If changing the SD card on the Pi 0 occurs, there's a strong possibility that the payloads on Foxtelem might no longer be counted. In some instances, it becomes necessary to perform a "delete payload," essentially rebooting the data page and creating a new data file within Foxtelem's source files. Consequently, in the event of executing a "delete payload," it's crucial to modify the Rasp.py script with the correct name if you intend to ensure continuous transmission for the real-time 3D visualization on the website.

---

5. Functionality issue.

- Camera.

During our recent tests, we discovered that the image capturing functionality with the Pi camera didn't work as seamlessly as outlined in the tutorial and continues to remain non-functional. Following the steps and without delving too deeply into the details, we integrated the camera. Our initial check involved verifying if Foxtelem correctly detected its presence, which it did, displaying "OK" rather than "Fail" as observed when the camera wasn't connected.

Following the tutorial, we switched to SSTV mode to transmit a photo every 30 seconds. However, while the QSSTV software detects communication, no image is transmitted. The issue doesn't seem to be the system's incapability to convert the .wav file into an image, as a test mode confirms that this functionality works perfectly.

The problem seems to lie in the communication of data from the Pi 0 to the ground station. The script responsible for image transmission is rpitx.py, which includes a section for sending or attempting to send images, whether the camera is present or not. Therefore, even in the absence of the camera, the Pi 0 should be transmitting images to the software. However, it fails to do so. When manually initiating the script and monitoring the print statements on a .log file, we notice that one image is successfully sent, but not the second one.

The issue likely resides within this section of the script. However, due to limited time before the end of our project stage, we opted to focus on other more critical tasks.

It is, however, important to specify that we performed these actions on cards whose code was sometimes modified for improvements. One of the recommendations in a README is to reboot the images if we observe communication but no image is being read. This issue might arise from a frequency offset. We haven't tested this hypothesis. (We also enabled the use of the camera, but nothing changed).

Several steps were taken, involving hardware checks, software commands, and debugging attempts. Unfortunately, despite these efforts, the establishment of a functional system was not achieved.

Verification of Camera Presence and Configuration:

The initial steps involved ensuring the presence and correct configuration of the camera on the Raspberry Pi. Commands such as 'raspistill' and 'vcgencmd get_camera' were used to verify the camera's existence and operational status. Permission adjustments, including adding the user to specific groups (such as the 'video' group), were also attempted to ensure appropriate access to the camera. Ultimately, for this step, the camera was indeed present and detected successfully.

Capturing and Manipulating Images:

The use of the 'raspistill' command to capture images with defined resolutions was undertaken to obtain the required image files. File manipulations, including file type verifications (JPEG or other formats), were performed to confirm the nature of the captured data. Ultimately, in this part, it was possible to capture images using the camera. Additionally, upon observing the "CubeSatSim" folder while it is in SSTV mode, it was noted that a photo is taken, followed by the conversion of this image from .jpeg to .wav, and subsequently, the deletion of the .jpeg file.

Conversion and Data Transmission:

Attempts were made to convert the captured images into audio files using audio processing tools such as 'csdr,' aiming for transmission via RF with the utilization of 'rpitx.' Essentially, we tried to replicate manually what the 'rpitx' file accomplishes. Debugging efforts were undertaken to resolve issues related to file conversions and format compatibility. (Messages in the .log file generated during the execution of 'rpitx' displayed inconsistent outcomes, frequently indicating errors regarding unrecognized jpg formats.) This involved checking file permissions to allow proper data manipulation. (As the .wav file created from the .jpeg didn't inherently possess all permissions, we attempted modifications within the source code by creating a copy of 'rpitx,' but none of these attempts were successful.)

Compilation and Execution of Source Code:

As part of the resolution attempts, understanding and executing a C language source code ('cubesatsim.c') were involved to display outputs in the terminal, aiming to comprehend any potential issues encountered during data transmission.

Despite these methodical and exploratory efforts, the system did not achieve functional image transmission to the ground station via RF. The challenges faced may have been influenced by file format compatibility issues, difficulties in data type conversions, insufficient file permissions, or other unidentified technical problems. Further in-depth investigations might be necessary to address these specific issues and achieve a functional solution for image transmission.

For me : The main issue appears to be with the transmission process itself (with the rpitx script), rather than with conversion or compatibility concerns.

End.

This report serves as a comprehensive documentation of the challenges faced and the innovative solutions applied during the CubeSat's construction and operational phases.

Nimbus Center
Cork Institute of Technology Campus, Bishopstown, Cork
(021) 433 5560