Implementation of a Framework for the On-Board Computer in the Cubesat Test Bench Project Hycube

Matthieu Basset[1, *]

[1] *Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31055, Toulouse, FRANCE*

(Dated: August 19, 2022)

***Abstract*— The lifetime of a Cubesat is often less than expected due to errors in the functioning period. These errors could often be avoided by testing more before launch. Then, developing a testing environment for Cubesats must be a priority in every project to ensure safety for the mission. The Hycube test bench is to fulfil that purpose. It will provide a framework for teams working on Cubesat projects to run simulations on the functioning of the satellite.**

Keywords*—*Cubesat, Nanosat, Framework

I. INTRODUCTION

While there is an exponentially increasing number of cubesat launched each year. A huge part is still failing to communicate with a ground station after deployment or become unavailable due to errors, making them space junk. Death on arrival (DOA) affects around 20% of Cubesats successfully launched. Even passed the first day, the reliability of Cubesat is still low [1].
Then working on testing satellites is crucial for Cubesat projects. This is the main motivation for the Hycube project: a test bench for Cubesats in ISAE-SUPAERO. It is used for now to test the Cubesat CREME: a cooperation between U-Space, ONERA and ISAE-SUPAERO, which goal is to measure radiations in the Van Allen belt. The aim of the test bench is to simulate the behaviour of the satellite as accurately as possible

II. CONTEXT

A. Cubesats/Nanosats

There are several categories to differentiate satellites. The two that are relevant to detail in this article are Cubesats and Nanosats. Nanosats are satellites that weight less than 10kg, they are a subcategory of smallsats that are satellites lighter than 500kg. Cubesats are satellites that have a given size, one unit (noted 1U) is a cube of 10×10×10cm with a weight of 1.33kg at most. Cubesats are described as multiples of a unit (1.5U, 2U, 3U, 6U, etc...).

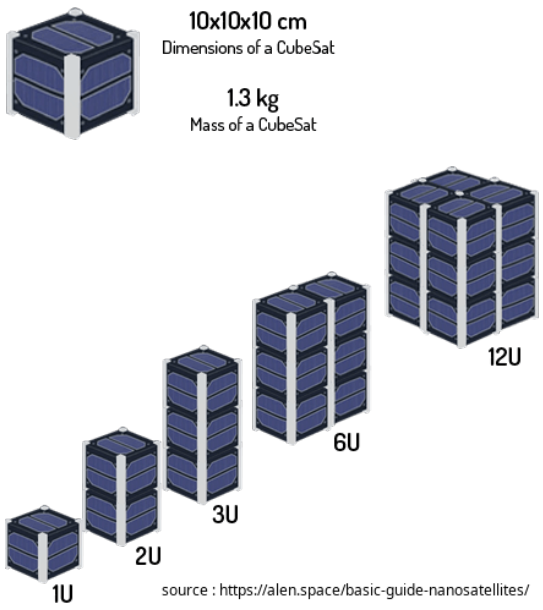

FIG. 1. Cubesats most common formats

A majority of the Cubesats are launched by universities or companies that are interested in the low cost of these projects that is appealing for various experiments in space.

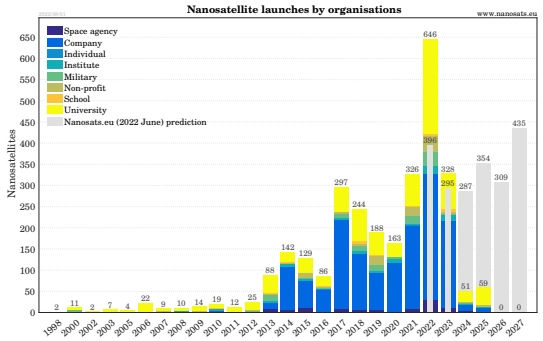

FIG. 2. Nanosats launched each year by organisation

* matthieu.basset@student.isae-supaero.fr

B. Satellites test benches

However, cubesats project tend to fail very often. Around 20% of cubesat launches result in DOA. Then it is crucial to test as much as possible cubesats while they are still on the ground. To that effect, a solution is doing flatsats: having all the components put on a test bench to conduct experiments on the satellite behaviour. The Hycube project is precisely focused on developing a test bench for the CREME cubesat and then is planned to be reused for future projects.

In our case, the test bench is separated in several testing subsystems:
- The main flatsat that simulates the behaviour of the satellite, with the real on-board Ninano card.
- Other card such as ZedBoard or Zybo to perform smaller scale tests without having to use the Ninano board.

C. Hycube structure

The Hycube test bench project is divided in several parts that interact with each other:
- The Mission planning that designs an initial planning for the mission.
- The Operation part that sends to the ground segment each task that the satellite has to do accordingly to the planning.
- The Ground Segment (GS) must send telecommands (TC) to the On-Board Computer (OBC) the tasks encoded in packets. It must also be able to telemeasures (TM) from the OBC.
- The OBC must receive TCs from the GS and give the right orders to the simulation or the Payload.
- The Payload is a camera that is used to take photos of the Earth. It is not used in the first iteration.
- The Simulation returns simulated measures to the OBC to for testing purposes.

The organisation is summed up in figure IV D

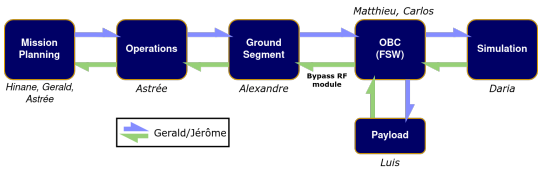

FIG. 3. Hycube project team organisation

D. CCSDS communication protocol

The Consultative Committee for Space Data Systems (CCSDS) is a forum that defines norms and give recommendations for satellites. Their guidelines are helpful when designing Cubesats. Their especially provide a format for packets and frames to exchange data [2] [3]. Since coding and decoding packets hasn't been implemented yet for the first iteration on the OBC side, they won't be detailed in this article.

III. AIM & OBJECTIVES

The main goal for my internship is to have a working first iteration of the Hycube project. I also have to prepare the cryptography part which shall be in the next iteration of the project.

A. Hycube first iteration

The first iteration's aim is to have a solid basing to work on for the rest of the project. The goal is to send information from the operation software on the ground to the simulated flight system. For the first iteration, a lot of complexity has been cut through:
- The operation will send only a few commands to the GS.
- The GS will send a packet instead of a frame to the OBC to simplify treatment.
- The OBC will not decode the packet but read the raw data and simply recognise some bits to figure out which command has been sent.
- The OBC will send a unique standard command to the simulation.
- The OBC will send back a 'ping' TM to the GS instead of a packet with the measure information.
- The Simulation will only send back time and altitude out of the data simulated.
- The Payload (camera) isn't used.

B. Cryptography

While the first iteration does not include cryptography, the second will, then preliminary research has to be done to facilitate future work. CREME project will implement Elliptic Curve Cryptography (ECC) to cipher TC and TM. In the second iteration of the Hycube project, the TCs will be ciphered by the GS, then deciphered by thee OBC. And the TM will be ciphered by the OBC to be deciphered by the GS after.

IV. METHODOLOGY

While working on a iteration based project, some concessions have to be done. One of them is not using Wireless communication with antennas to exchange information between GS and OBC. Then to exchange data, the first iterations will use ethernet cables to exchange TCP packets.

A. Python

An important part of the work performed to interface the OBC with GS and simulation has been done in python. There are several reasons to justify this choice. First, python is simpler to use and to work on for a team than C code. Then it has library to deal with TCP packets without worrying about low level packet reading. Also, the ZedBoard is used with UART protocol described in IV B that has a support with the `serial` library in python.
The code is modular so the tests can be run easily, several modes are implemented in different scripts for each use case. All the configuration for the tests use JSON files to ease the user usage.

B. Embedded

The embedded part is written in the C programming language, moreover, it doesn't use dynamic memory allocation since the allocation time isn't deterministic, which isn't suitable for a Cubesat use. It also doesn't use most of the standard C library since most of it isn't used and would be too heavy for the limited amount of memory of the satellite.
The code is sent to the card using the Xilinx software development kit by AMD.
The communication with the board uses the UART protocol which is a simple protocol to transfer data with embedded systems via USB.

C. Cryptography

1. ECC

Elliptic Curve Cryptography (ECC) is based on the Elliptic Curve Discrete Logarithm Problem that is known for being extremely slow to reverse [4]. It uses elliptic curves on finite (or Galois) fields $\mathbb{F}_p$.
The seed is a point on an elliptic curve, then by multiplying this point k times, another point is obtained, this point is the public key and the number k is the private key.
The ECC is used as a symmetric ciphering process for this project by exchanging public keys to form a shared secret.
While Rivest-Shamir-Adleman (RSA) ciphering is widely used, for a Cubesat use, the Elliptic Curve Cryptography (ECC) is preferable. Indeed, for the same amount of protection, the key size is small by a factory up to 1:30 for 256 bits of security.

Security	≤ 80	112	128	192	256
ECC Key Size (bits)	160	224	256	384	521
RSA Key size (bits)	1024	2048	3072	7680	15360
Key size ratio	1 : 7	1 : 10	1 : 12	1 : 20	1 : 30

TABLE I. RSA key size vs ECC key size for different levels of security

2. Implementation

The mbedtls library [5] enabled the actual implementation of the ECC cryptography in the embedded code. It is a lightweight library that implements the basic of cryptography in the C programming language.
The previous intern working on the subject has already adapted the mbedtls library to the project by removing parts of the source code useless to the project. And I fixed the last compilation issues to ensure the correct operation of the library.

D. Interfaces

The interfaces for the first iterations were therefore limited to the following ones.

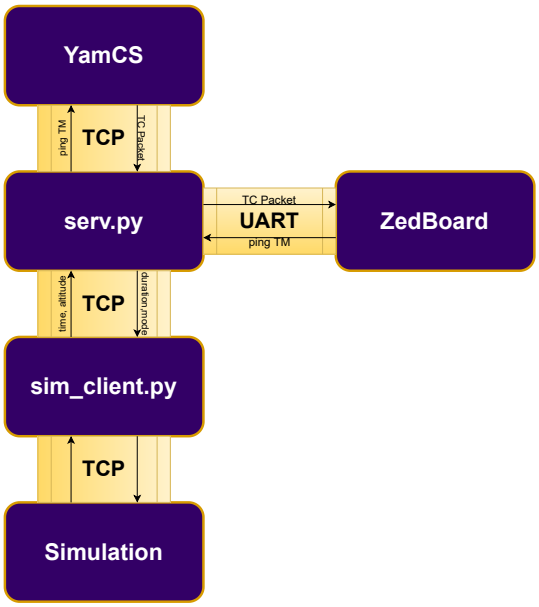

FIG. 4. The interfaces of the OBC for the first iteration

V. RESULTS

A. Interfaces

For the demonstration on July the 21st everything worked as planned, the packet sent from the operation

software got to the GS that transmitted it to the OBC. The OBC sent the command to the simulation that gave the result back. The result has been printed on the terminal running the python interface script. And finally the ping TM was sent to the GS. The framework is working as expected and is waiting for the second iteration to be refined.

B. Ciphering library

The library successfully generates private and public keys. It can also generates certificates with all information provided.

VI. DISCUSSION

A solution to bypass the use of UART is to configure the board's ethernet port to allow transfers via TCP directly to the board. I didn't go in much details to try to make it work since it was more on the hardware part of the development, but it will be useful on the second or third iteration of the project.
Since the cryptography wasn't the highest priority for the first iteration, there is still some work to do on it, especially writing or get a library to have a ciphering function. Indeed in its current state, the library can generate private and public keys and also certificates, but cannot use them to cipher an information.

VII. CONCLUSION

Developing a framework to interface every part of an IT project represents almost as much time as developing each part individually, it needs a lot of discussions to make sure everyone is okay with the definition of the data to exchange. And the earlier is always the better to have these talks, because late definition of interfaces often end up in huge project refactoring. Then working on the developing of a framework for each Cubesat project

A. Future work

As explained in VI the board shall use an ethernet driver to simplify all communications and in the long term getting rid of the python part.
The second iteration will also begin shortly after the success of the first one, then a lot of work around cryptography will have to be done. The interaction with the camera must also be handled, the saving process of the picture and sending it to the GS.

VIII. ACKNOWLEDGEMENT

Thanks to Thibault Gateau, my tutor, for guiding me through the subject in a very limited time, to Marc Justicia-Mayoral for his precious help and expertise on the embedded code. And thanks to the whole CREME team for their cooperation in the project by sharing all needed information and helping me on various aspects.

[1] M. Langer and J. Bouwmeester, Reliability of cubesats - statistical data, developers' beliefs and the way forward (2016).
[2] C. C. for Space Data Systems, Space packet protocol (2020).
[3] C. C. for Space Data Systems, TM space data link protocol (2021).
[4] S. Nakov, Elliptic curve cryptography (ECC) (2018).
[5] TrustedFirmware, MbedTLS github repository (2022).