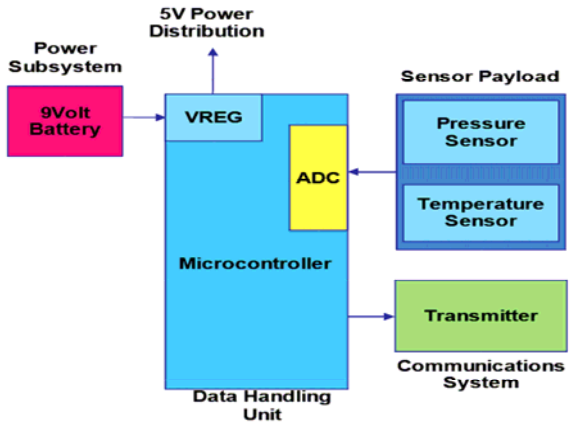
ECESAT: a CanSat to analyze the topology of earth

Quentin Allouche, Julie Cristofol, Christelle Reanaud, Aditya Raj, Sarmila Thanarajah

I. Introduction

'T'HE ECESAT is the first Cansat design by ECE students. The various missions handle by the device are the telemetry of the atmosphere by finding the temperature and the pressure. The Cansat has also to realize a "come back mission", which means it has to fly back to the point of origin (where it started). Finally; it can also capture some images and do image processing by recreating the geography of the earth.



II. Context of Development

A. Club

The group is composed of five students from major Embedded System and Transports & Mobility. The composition of the group will allow us to easily manage the mechanical and electronic part of the project.

Quentin ALLOUCHE is the project manager and will follow up the different stages of the project. The team under Quentin is divided into two subgroups: one of them is dealing with the communication part, and the other one with electronic part.
Indeed, Quentin ALLOUCHE and Julie CRISTOFOL work together on the communication part between the CanSat and the ground station and on the parachute. The other subgroup composed by Aditya RAJ, Christelle RENAUD and Sarmila THANARAJAH will handle the electronic part of the project with the design of the circuit board.

B. Work plan

The project duration was one year. The first four months were spent on researches on the existing Cansats: What it actually is? How to make one? How does it work, the basic principles? What are the constraints and all the payloads used? And so on. At the end of the last month, we ordered the different components of the Cansat to realize the hardware part. Then the fifth month, we started the software part and the assembly of the different hardware part. Finally for the last months, we did all the tests to complete the ECESAT. The team was divided in two parts: the hardware part and the software part. In the hardware part, we have designed the board and then placed the different modules and written the code to make it works for 3 weeks. In the software part, we have designed the user interface during one week to send some parameters to the CanSat and to display the pressure, the temperature or pictures of earth retrieved by the system. Then the image processing to recreate the surface of earth took 2 weeks.

III. Definition of the Missions

The goal of the team is to create a landing module the size of a soda can (66mm diameter and 115mm height). The dimensions will be checked on the day of the competition thanks to a template in which the mini satellite will be content to be valid. It also cannot exceed the weight of 350gr. It is an off-line system that performs scientific missions, which will be automatically dropped from a balloon or a rocket at a defined altitude (between 100 and 150 meters).

The team has to choose missions proposed by the CNES on the specification document. The proposed missions are:

- Atmospheric sounding: temperature, pressure, etc.
- Development of an XBee Communication: this mission will allow the communication between the concept and the ground station.
- Photo/video
- Airbag landing; the mini satellite should go back to the ground thanks to a parachute but should also be fitted with airbag to soften shocks.

Detailed design of components

Here is the general window of the GUI, when the application is executed with Visual Studio 2008:

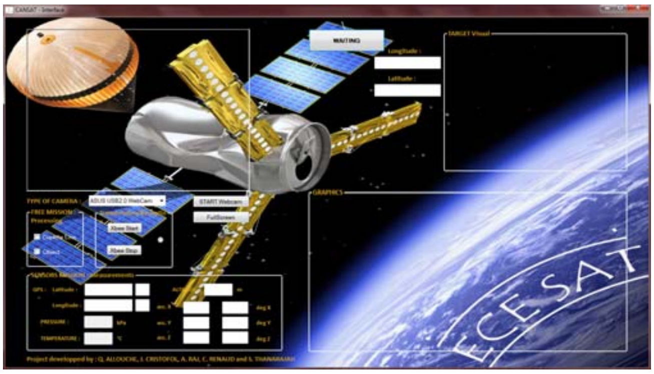
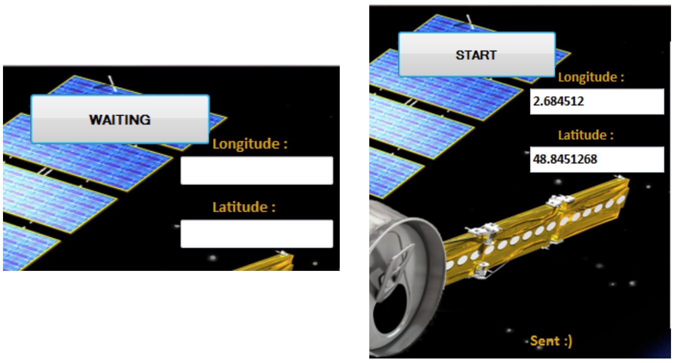


Figure 1: IHM on ground station

This class “Windows Form” is called “CANSAT Interface”, and is always visible and placed automatically in the center of the screen. Its dimension is 1366x768 pixels. The user cannot modify the size of this window, the size is fixed. If the user closes the general window, the entire program stops.



2.684512

48.8451268

Figure 2: GPS coordinate of the target entering and sending

The "Start" button allows to start displaying the received data and the received video stream, when you can click on it, with the method VideoCaptureDevice () which registers the driver for the selected camera and NewFrameEventHandler () which updates the video.

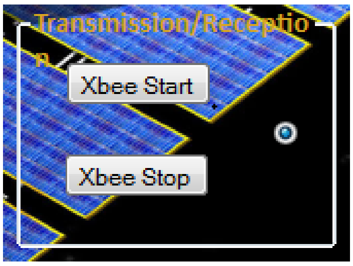


Figure 3: Xbee communication module

By default when the user clicks the button “Start”, the “radioButton” is checked which means that the Xbee communication works. The buttons “Xbee Start” et “Xbee Stop” can stop or restart the wireless communications and the “RadioButton” is there to notify visually the state of the communication.

Dimensions and place of the "PictureBox" are defined in the code during initialization. This "PictureBox" will allow us to display the video stream recovered from CanSat live.
We chose to show image processing in real time on the same "PictureBox" for stream video to have more places to display other components.

The "ComboBox" displays the list of all video devices installed on the computer by default, to allow the user to choose the camera that he wants to use. For this, we use the methods FilterInfo () and FilterCategory () of libraries Aforge. In our case, the user chooses the onboard camera on CanSat.

The “Button” FullScreen proposes to the user to display the video stream in another window, automatically centered and of modifiable size, where the user can see the original video or the video with processing of images (explained later) in full screen in real time.

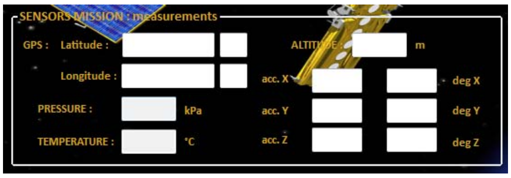


Figure 4: Sensor measurements received from the can

The dimensions, the place, the text of the 15 “labels” are initialized in the declaration of variables and won’t change during the execution.
And the dimensions and the place of the 13 “textBox” won’t change but the contents “NomDuTextBox.Text() of each of “textBox” will be updated every 1 ms by the presence of a timer that starts from the beginning of the execution. The text displayed in the "textBox" is the measure corresponding to a sensor.

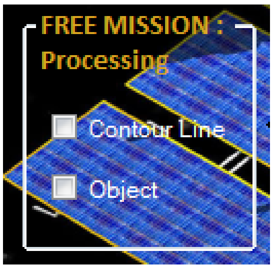


Figure 5: Processing

For this part, we use library Emgu to execute this king of processing anf follow some tutorials. All the algorithm is explain in the program.

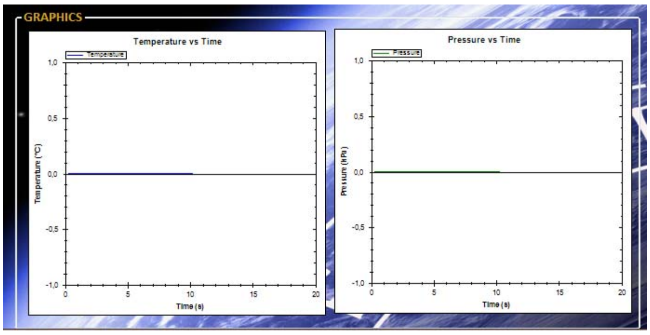
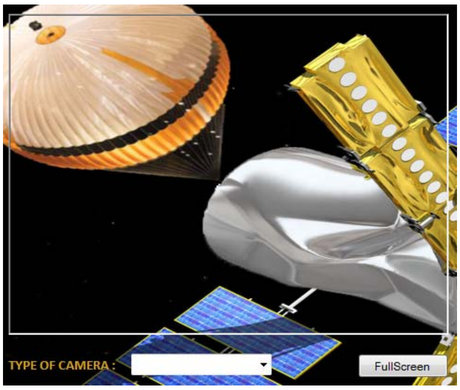


Figure 2: Graphics of measurements



In this part, we display graphs from the received data in real time with the library ZedGraph.

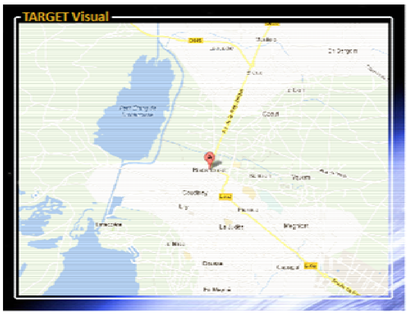


Figure 7: Target on map

Once the user click on the button “Waiting” after entering GPS coordinate of the target, this coordinate will be displaying with a red point. Then, when the can will be in the area of the target, we will see the evolution of the direction of the CanSat.

A. *Scientific Mission*

The one and only Goal of CANSAT is to send telemetry data successfully. That is possible only when all the sensor system is working perfectly and the transmission is valid.

Well each and every action performed has an equal reaction, which tends to complement the whole system and make our satellite work.

The temperature sensor senses the temperature with the help of LM35, which in turn is used to calculate the height. The pressure is also attained using the sensor. And that in turn also can be used for attaining altitude value using the expression mentioned. Further the Camera module with the help of compass for orientation is used to capture images as per mission requirement and sent to Ground STATION;

The accelerometer is used to calculate the ground impact while landing. Well the landing is controlled by descent control mechanism, which in our case is a parachute. GPS is also used for coming back to exact location given by the jury before the competition.

B. *Free Mission*

One of the missions, the free one, is to recreate the relief from a satellite image. The main idea is to convert the image to treat in level of gray and to pick up the value of the pixels and to plot them on a 3D graph to recreate the relief. Thus we developed an algorithm to make such a processing (8):

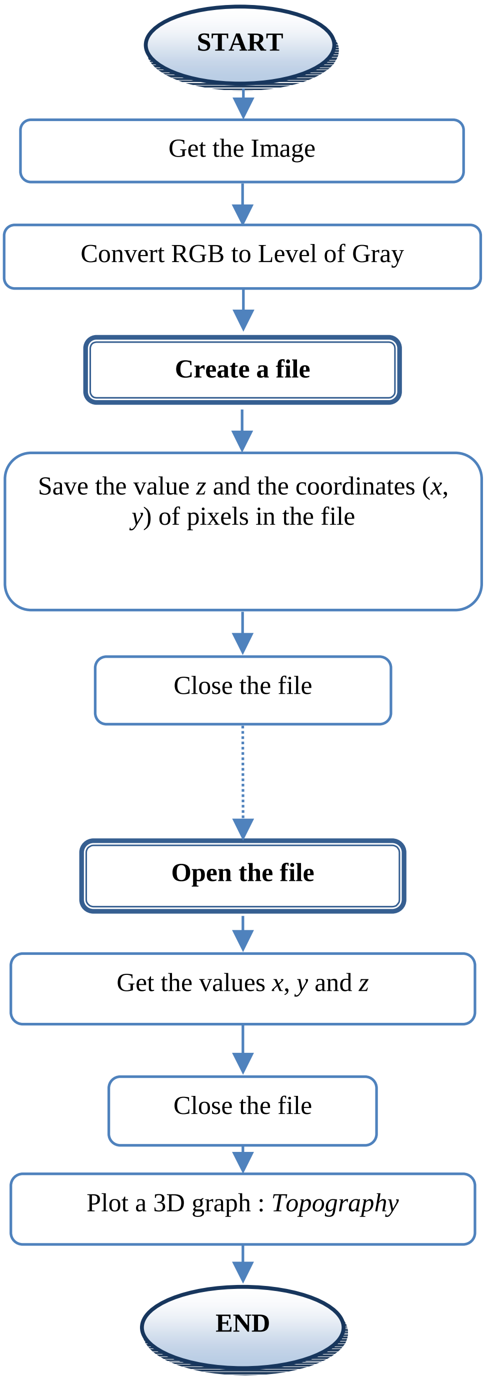

START
Get the Image
Convert RGB to Level of Gray
Create a file
Save the value z and the coordinates (x, y) of pixels in the file
Close the file
Open the file
Get the values x, y and z
Close the file
Plot a 3D graph : Topography
END

Figure 8: General algorithm of the software structure

The interface of the processing consists in a Windows Presentation Foundation (WPF) application which appeals to a Windows Form (WF) application developed in C# with Visual Studio 2008. The first one is needed to add a viewport3D to host the 3D topography with a camera and the second one is used to load and print the satellite images.

Conversion in level of gray

Once the picture is loaded, it is immediately converted in level of gray thanks to the use of the library *EmguCV* based on *OpenCV* which is mainly dedicated to image processing.

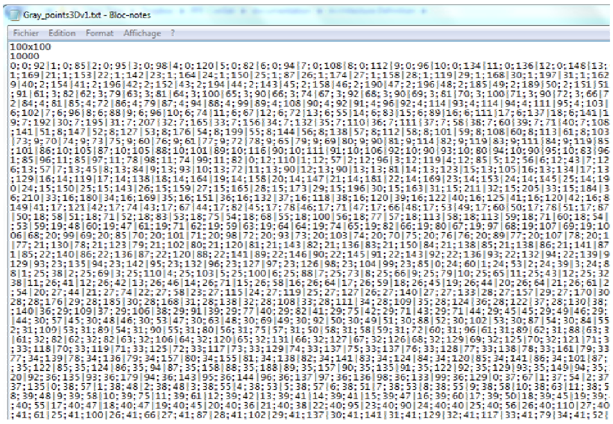
Save and salvage the data

The position in coordinates and the value of the pixels of the image converted in level of gray are stocked in a table and then saved in a file *.txt*. It also contains the dimension of the image and the numbers of pixels. This file allows the user to keep a record. Indeed for scientific missions it is always more sensible to keep records to make any sorts of processing and to compare the results from all of the attempts.

Thus we take care not to erase files when another saving is required.

We also take care to not open a file which does not exist thus avoiding any problem with the software:

Once the file is open, the dimension and the number of the image are first fetched and then the values for recreating (9):


Gray_points3Dv1.txt - Bloc-notes
Fichier Edition Format Affichage ?

```
100x100
10000
0;0;92|1;0;85|2;0;95|3;0;98|4;0;120|5;0;82|6;0;94|7;0;108|8;0;112|9;0;96|10;0;134|11;0;136|12;0;148|13;
1;169|21;1;153|22;1;142|23;1;164|24;1;150|25;1;87|26;1;174|27;1;158|28;1;119|29;1;168|30;1;197|31;1;162
9|40;2;154|41;2;196|42;2;152|43;2;194|44;2;143|45;2;158|46;2;190|47;2;196|48;2;185|49;2;189|50;2;151|51
;91|61;3;82|62;3;79|63;3;81|64;3;100|65;3;90|66;3;74|67;3;92|68;3;90|69;3;81|70;3;100|71;3;90|72;3;66|7
2|84;4;81|85;4;72|86;4;79|87;4;94|88;4;99|89;4;108|90;4;92|91;4;96|92;4;114|93;4;114|94;4;111|95;4;103|
6;102|7;6;96|8;6;88|9;6;96|10;6;74|11;6;67|12;6;72|13;6;55|14;6;83|15;6;89|16;6;111|17;6;137|18;6;141|1
9;7;192|30;7;195|31;7;207|32;7;165|33;7;156|34;7;132|35;7;110|36;7;111|37;7;58|38;7;60|39;7;71|40;7;108
;141|51;8;147|52;8;127|53;8;176|54;8;199|55;8;144|56;8;138|57;8;112|58;8;101|59;8;108|60;8;113|61;8;103
|73;9;70|74;9;73|75;9;60|76;9;61|77;9;72|78;9;65|79;9;69|80;9;90|81;9;114|82;9;119|83;9;111|84;9;119|85
;101|86;10;105|87;10;105|88;10;101|89;10;116|90;10;111|91;10;106|92;10;90|93;10;80|94;10;90|95;10;83|96
1;85|96;11;85|97;11;78|98;11;74|99;11;82|0;12;110|1;12;57|2;12;96|3;12;119|4;12;85|5;12;56|6;12;43|7;12
6;13;57|7;13;45|8;13;84|9;13;93|10;13;72|11;13;90|12;13;90|13;13;81|14;13;123|15;13;105|16;13;134|17;13
;129|16;14;119|17;14;138|18;14;164|19;14;158|20;14;147|21;14;181|22;14;169|23;14;153|24;14;145|25;14;19
0|24;15;150|25;15;143|26;15;159|27;15;165|28;15;173|29;15;196|30;15;163|31;15;211|32;15;205|33;15;184|3
6;210|33;16;180|34;16;169|35;16;151|36;16;132|37;16;118|38;16;120|39;16;122|40;16;125|41;16;120|42;16;8
149|41;17;121|42;17;74|43;17;67|44;17;82|45;17;78|46;17;71|47;17;66|48;17;53|49;17;60|50;17;78|51;17;87
|50;18;58|51;18;71|52;18;83|53;18;75|54;18;68|55;18;100|56;18;77|57;18;113|58;18;113|59;18;71|60;18;54|
;53|59;19;48|60;19;47|61;19;71|62;19;59|63;19;64|64;19;74|65;19;82|66;19;80|67;19;97|68;19;107|69;19;10
06|68;20;99|69;20;85|70;20;101|71;20;98|72;20;93|73;20;103|74;20;70|75;20;76|76;20;89|77;20;107|78;20;1
|77;21;130|78;21;123|79;21;102|80;21;120|81;21;143|82;21;136|83;21;150|84;21;138|85;21;138|86;21;141|87
1|85;22;140|86;22;136|87;22;120|88;22;141|89;22;146|90;22;145|91;22;143|92;22;136|93;22;132|94;22;139|9
129|93;23;135|94;23;142|95;23;132|96;23;127|97;23;126|98;23;104|99;23;85|0;24;60|1;24;53|2;24;39|3;24;8
8|1;25;38|2;25;69|3;25;110|4;25;103|5;25;100|6;25;88|7;25;73|8;25;66|9;25;79|10;25;65|11;25;43|12;25;32
38|11;26;41|12;26;42|13;26;46|14;26;71|15;26;58|16;26;64|17;26;59|18;26;45|19;26;44|20;26;64|21;26;61|2
;54|20;27;44|21;27;74|22;27;58|23;27;115|24;27;119|25;27;127|26;27;140|27;27;133|28;27;157|29;27;170|30
28;28;176|29;28;185|30;28;168|31;28;138|32;28;108|33;28;111|34;28;109|35;28;124|36;28;122|37;28;130|38;
;140|36;29;109|37;29;106|38;29;91|39;29;77|40;29;82|41;29;75|42;29;71|43;29;71|44;29;45|45;29;49|46;29;
|44;30;57|45;30;48|46;30;53|47;30;63|48;30;69|49;30;92|50;30;49|51;30;88|52;30;102|53;30;87|54;30;84|55
2;31;109|53;31;89|54;31;90|55;31;80|56;31;75|57;31;50|58;31;58|59;31;72|60;31;96|61;31;89|62;31;88|63;3
|61;32;82|62;32;82|63;32;106|64;32;120|65;32;131|66;32;127|67;32;126|68;32;129|69;32;125|70;32;121|71;3
;33;118|70;33;119|71;33;125|72;33;117|73;33;129|74;33;137|75;33;137|76;33;128|77;33;138|78;33;161|79;33
77;34;139|78;34;136|79;34;157|80;34;155|81;34;138|82;34;141|83;34;124|84;34;120|85;34;141|86;34;101|87;
;35;122|85;35;124|86;35;94|87;35;158|88;35;188|89;35;157|90;35;135|91;35;122|92;35;129|93;35;149|94;35;
20|92;36;135|93;36;179|94;36;143|95;36;144|96;36;137|97;36;136|98;36;133|99;36;129|0;37;67|1;37;54|2;37
37;135|0;38;57|1;38;48|2;38;48|3;38;55|4;38;53|5;38;57|6;38;51|7;38;53|8;38;55|9;38;58|10;38;63|11;38;5
8;39;48|9;39;58|10;39;75|11;39;61|12;39;42|13;39;41|14;39;41|15;39;47|16;39;60|17;39;50|18;39;45|19;39;
;40;55|17;40;47|18;40;47|19;40;45|20;40;36|21;40;38|22;40;95|23;40;90|24;40;40|25;40;56|26;40;110|27;40
;41;61|25;41;100|26;41;66|27;41;87|28;41;102|29;41;137|30;41;141|31;41;129|32;41;117|33;41;79|34;41;52|
```

Figure 9: Extract of the saved file named Gray_points3Dv1.txt.

The character "|" for splitting allows to get one by one the value of the pixel with its coordinate and to put them into the table of Point3D. We also take care to convert to double the values fetched because from a *.txt* we only get String.

Recreate the relief

The relief is recreated from the data fetched on the last saved file.

The code for this part is developed thanks to an example code from 3D graphic tutorial available on the internet.

A. *Software Behavior for user*

In this part we detail all the user handling of the processing software.

First the user sees appearing the WPF window when he starts the application for processing. It contains thee buttons: "LOAD", "TOPOGRAPHY" and "QUIT".

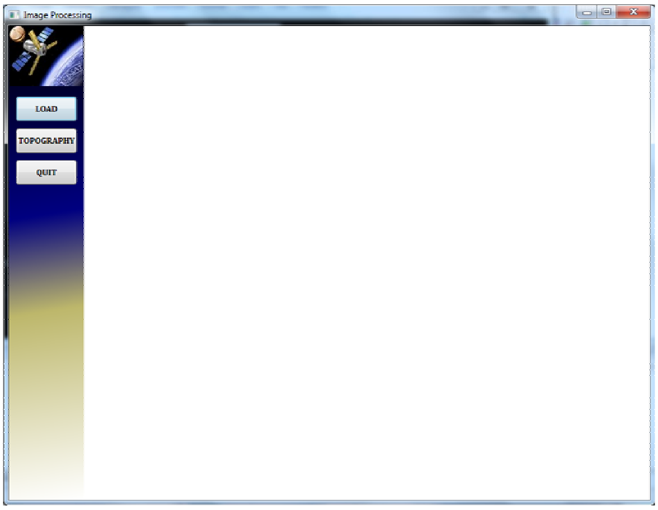
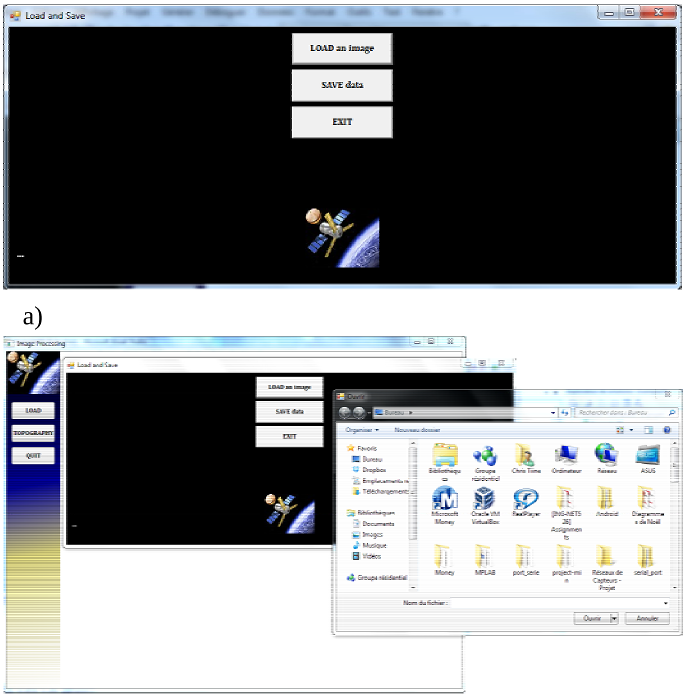


Figure 3: Opening window of the processing software.

Clicking on the button "LOAD" opens another window, a WF window, (Figure 4 a.). This one contains three buttons too: "LOAD an image", "SAVE data", and "EXIT". The user can fetch the desired satellite image to process (Figure 4 b.). Once it is loaded, it will be immediately converted in level of gray and visible on the screen.



a)

Figure 4: Opening of the loading window; b) Loading image.

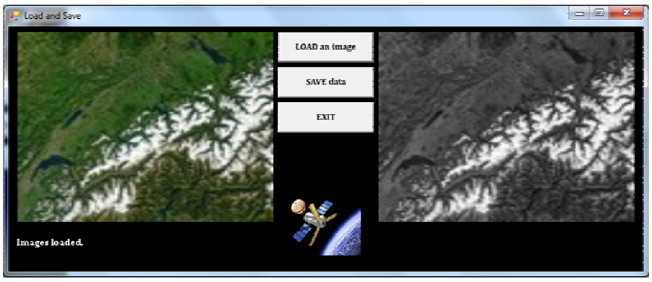


Figure 5: "Load and Save" window with printed images.

Before quitting the window of loading, the user has to save the data needed to recreate the relief by clicking on the button "SAVE data".

Once the user quits the "Load and Save" window, he comes back on the WPF window. He also can click on the button "TOPOGRAPHY" and has to wait some seconds to get results (Figure 6).

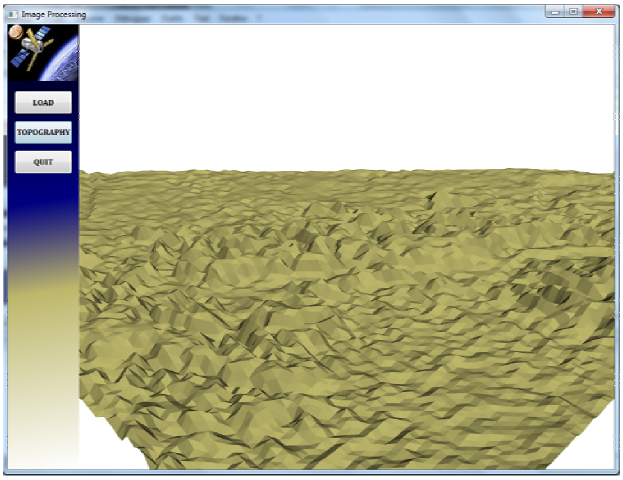


Figure 6: Recreating of the relief.

The position of the camera and the direction of the view are predefined by the developer.

IV. CanSat Architecture

A. *Electrical architecture*

The architecture of our CanSat is basically composed of four major subsystems:

- Sensor system
- Communication system
- Power system (Electrical and Power System: EPS)
- Processor system (Command & Data Handling: C&DH)

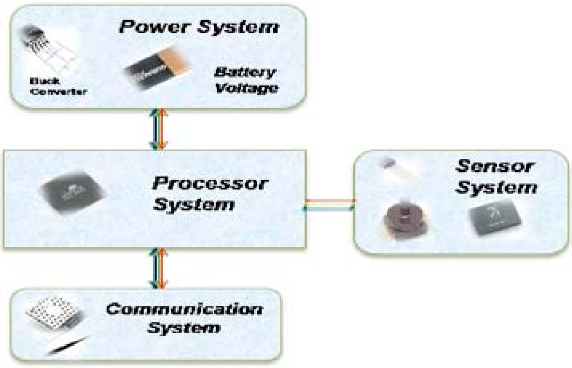


Figure 0 : Architecture of the CanSat

B. *Mechanical parts*

To locate the CanSat during its descent, we will use the module GPS OEM subminiature "EM-406'' with integrated amplified antenna (**Erreur ! Source du renvoi introuvable.**).

The format of the output message obeys the NMEA* protocol. All the frames are made as follow:

$The name of the frame (5 characters),data,data,…,data*hhCRLF

They always begin by the character "$" and always end by hh = Checksum.

CR: "Carriage return" (ASCII code 13)

LF: "Line Feed" (ASCII code 10)

The following accelerometer will be used to determine the landing force.

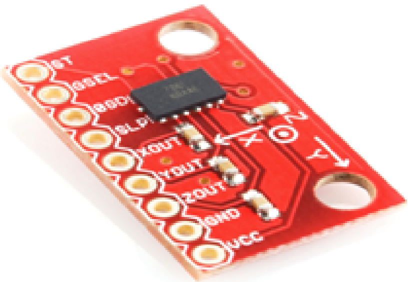


Figure 11: MMA7361L

It needs to sense the acceleration in all three dimensions to find the net acceleration.

To lead the Cansat, according to some computations done thanks to parameters retrieved by the accelerometer and the gps, one Micro-servo motor of 9g and operating at 5v would be mounted below the camera support module by screws.

Figure 12: The micro-servo motor

Camera mounted on top in the support module giving it 2 degrees of motion to rotate in the enclosing provided.

C. *Telemetry*

The XBee-PRO RF Modules was engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between devices.

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

The format is not predefined by the Xbee module. It might be defined by the user according to the data he wants to get. For our project we need to get latitude, longitude, altitude, temperature, pressure, direction. We also have the next frame format:

Temperature-Pressure-Latitude-Longitude-Altitude-Direction

V. Conclusion

At this point, the project is at ¾ of it s term. Some design of the board and some tests remain to finish the CanSat.

In order to finish the project some appointments the weekend will be done.