CFP-18-Ursa Minor

CanSat Final Paper

Stavros Panos
1o EPA.L. Prevezas, Greece, prevezagreece13@gmail.com

Theodoros Kakiouzis
1o EPA.L. Prevezas, Greece, theodoros2001gr@gmail.com

Konstantinos Pappadias
1o EPA.L. Prevezas, Greece, papadiaskonstantinos@gmail.com

Athanasios Kordatzis
1o EPA.L. Prevezas, Greece, thanosk2001@gmail.com

Panagiotis Zisis
1o EPA.L. Prevezas, Greece, zisispanos87@gmail.com

Thomas Vasiliadis
1o EPA.L. Prevezas, Greece, nickproman8@gmail.com

Dimitrios Vasilakos
1o EPA.L. Prevezas, Greece, dhmhtrhsvasilakos@gmail.com

Abstract: Our team built a vehicle for the *CanSat in Greece* competition that was launched and executed both missions at a rather high level of success. While falling, the satellite sent and recorded air pressure and temperature, humidity, changes in the magnetic field, the position of the vehicle in space, geographical coordinates, but failed to store in its internal memory sd a couple of pictures taken from high up, due to our laptop malfunctioning. The recovery system functioned normally and the fall was smooth, without any instability or swirling. When Cansat landed, the outer shell (Lander) opened, and the rover came out. The wheels are expandable and the moment they expanded, their diameter was doubled. Following telemetry commands, the vehicle was able to move and take extra pictures the ground. All the pictures were sent through the telemetry channel. The remote controlling of the vehicle was conducted from the ground station, using a laptop that ran a GUI program. Making use of the measurements taken, we calculated altitude from air pressure, lift and fall velocity and created graphs on a spreadsheet, which present changes in temperature, humidity and magnetic field as relates to altitude.

I. INTRODUCTION

Cansat missions simulate a real satellite mission, but a satellite that is the size of a can.

The microsatellite was 'launched' via a drone at an altitude of about 500m where it was let to a free fall. At the same time a cross type chute opened up and our satellite descended at a velocity of about 8m/sec.

As its secondary mission, our team decided to build a rover that is equipped with a range of sensors such as: air pressure, temperature, humidity, gyroscopes, accelerometers, magnetometer and Global Positioning System (GPS). It is also equipped with a micro camera according to the Joint Photographic Experts Group (JPEG) protocol, in order to take pictures.

Our *Cansat* communicates with the ground station via a telemetry system at Ultra High Frequencies (UHF). It is must be pointed out that this is an advanced telemetry since we can send lots of commands to the satellite for it to execute an action as well as to monitor its status in real time. Moreover, the photos and all measurements stored in the Secure Digital (SD) card can be sent to the ground station through its unique narrow band telemetry channel. Data are sent in variable sampling rate, from one sample per 3 sec to 5 samples per sec.

Our *CanSat* is placed inside a flexible plastic shell, which protects the microsatellite and also keeps closed the expandable wheels of the rover. The chute is tied to the exterior shell. After landing and following a command from the ground station, the exterior shell opens up and the vehicle expands in its full size. The vehicle is able to move on quite rough terrain and send measurements and photos to the ground station.

II PROJECT DESCRIPTION

As shown in the following block diagram, our system consists of these parts :

A) Cansat

- The sensors materializing the two missions, such as pressure, temperature and humidity.
- The 3 axes magnetometrer, gyroscope and accelerometer. With these we measure the planet magnetic field strength and locate the vehicle in 3D space.
- One GPS module to determine the geographical position.
- The JPEG camera for taking pictures.
- Sensors to control the vehicle status, such as its battery level or informing about wheel blocking in the case of its bumping onto an obstacle.
- The microSD card to store measurements data and photos.
- The UHF transceiver to communicate with the ground station.
- Two *micro servos* for moving the *rover*.
- The lander release thermal element.
- The Micro Controller Unit (MCU) that controls everything inside the CanSat.

B) Ground station

- One directional UHF antenna.
- The UHF transceiver to communicate with the cansat.
- The MCU that prepares data before sending them to the laptop.
- The laptop controlling and navigating the cansat and monitoring its status.

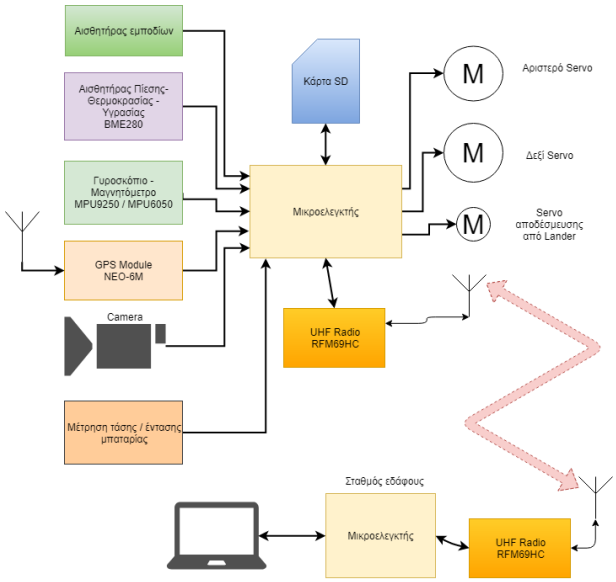

Fig. 1: The block diagram of the system. In the final revision the release servo has been replaced with a thermal element.

The exterior shell is kept closed by a nylon thread. The thread melts with the help of a resistor and the exterior shell opens up following landing.

Two independent micro servos make the wheels spin.

In the interior of the chassis we opted for a horizontal layering with three perfboards. In the middle one there is **Ursuino**, a DIY cansat controller.

The SD card, the sensors, the gyroscopes, accelerometers, the GPS module and the UHF transceiver are placed on the other two perfboards.

Communication is half duplex and uses the UHF Industrial Scientific and Medical (ISM) band, specifically at 434.000 Mhz.

All data from and to the satellite pass through this telemetry channel. The bitrate is 38400bps. The cansat antenna is an interior metallic rode, 8.4cm long, that is shorter than λ/4 (17cm), with impedance matching coil (Rubber).

On the ground station side, there is one arduino micro with the UHF transceiver that is connected to the laptop through a Universal Serial Bus (USB) port. On the ground station we used a directional Quagi type antenna with 12dbi gain. On the laptop a Graphic User Interface (GUI) program runs, through which we control the cansat.

II.I Materials and structural design

The mechanical structure is handmade. The materials can be easily found.

II.I.I The CanSat chassis

The body (chassis) of the cansat was made from Polyvinyl Chloride (P.V.C.), a durable and light material. The two servo motors are mounted on two PVC octagons. Two metallic rodes 3mm in diameter hold the octagons together. One rectangle PVC board is placed parallel to the rodes and vertical to the octagons. On the one side of this board, there is the first perfboard mounted, while on the other there are the two batteries (Fig. 2).

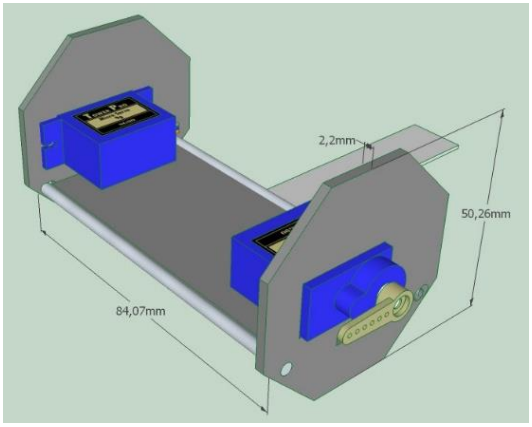

Fig. 2: The vehicle body - chassis (3D model)

Also, the tail that contributes to the right movement of the vehicle (avoiding the turning round of the body itself) is necessary.

The metal tail is mounted on the rectangle board and it is made of steel tape so that it can be folded inside the lander. The lower perfboard is mounted on the horizontal rectangle board and the other two are connected with connectors (Fig. 3).

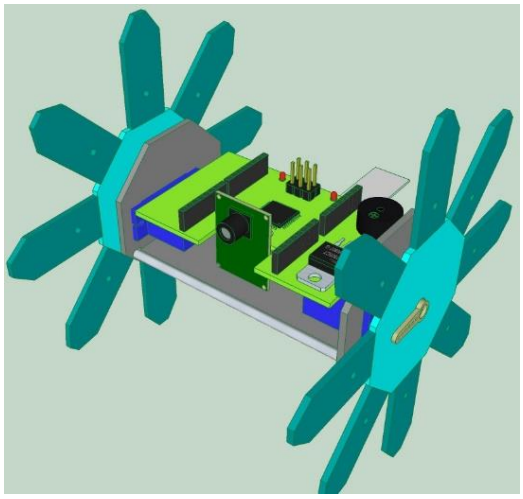

Fig. 3: Perfboard arrangement inside the vehicle (3D model)

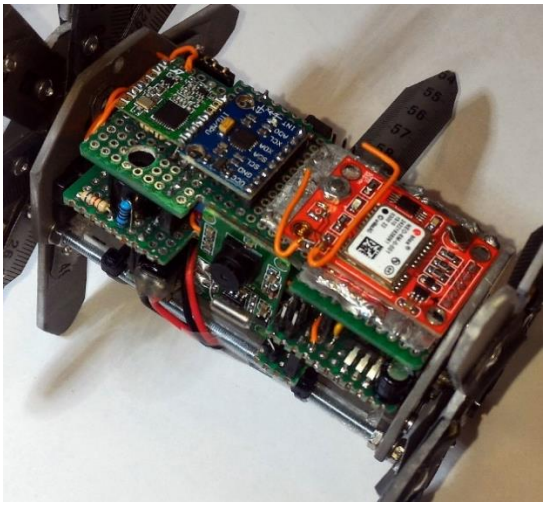

Fig. 4: End result.

II.I.II The wheels of the vehicle

Our vehicle has two wheels with expandable spokes. They are 58 cm in diameter when folded. With the help of steel tape, they expand and reach 125mm in diameter. The idea came from a rover used in the American competition (ref. 1).

The expandable wheels are also made of PVC. Two octagons are here as well. Each side has a PVC spoke connected. As a connector we used steel tape, so that it can be folded and then expand when the lander opens up (Fig. 5, 6).

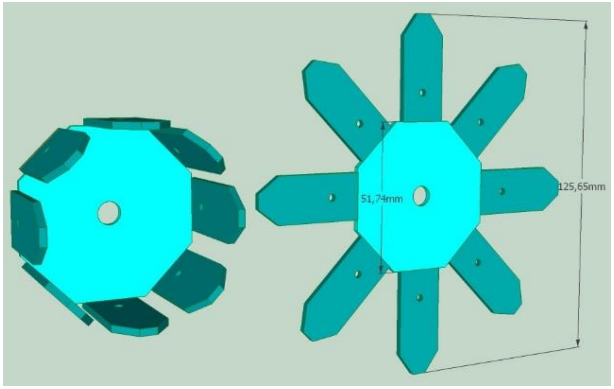

Fig. 5: Folded and expanded wheel (3D model).

The steel tape and the PVC parts are connected with rivets on the inside and rubber on the outside.

Fig. 6: End result. We can see the steel tapes taken from a measuring tape and the rivets.

II.I.III The exterior shell

As it was mentioned above, the vehicle is placed in the exterior shell, which is made of plastic, 0.4 mm thick. Once the wheels and the tail are folded, the vehicle is enclosed in the plastic in a specific way. There are openings, one in front for the camera and one in the back for access to the main switch and the SD card. (Fig. 7, 8, 9). When folded, it is hold by a nylon thread, that touches the thermal element in a specific way. The chute is tied to the exterior shell on two points so that the cansat descends horizontally and the camera faces sideways and downwards.

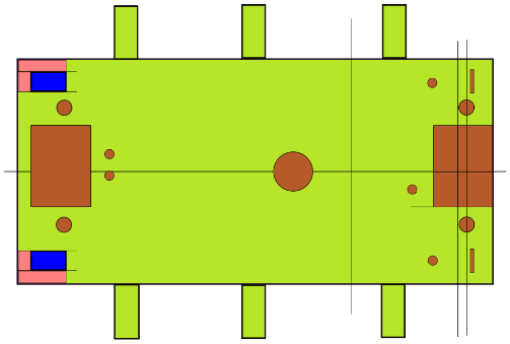

Fig. 7: Building pattern of the lander.

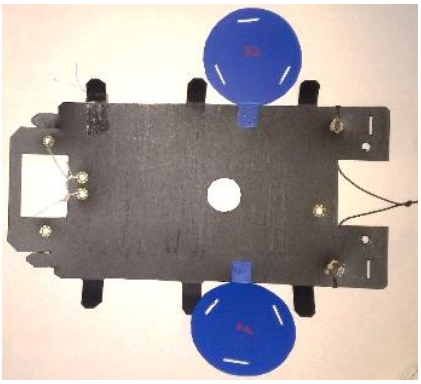

Fig. 8: The expanded lander prior to inserting the rover.

Fig. 9: The lander ready for the rover to be inserted.

II.II CanSat Electronic components and electrical design

In the diagram below (Fig. 10) all the components of the cansat and their interconnections are presented. There are components that function at 5V and some others at 3.3V as shown in Table 1.

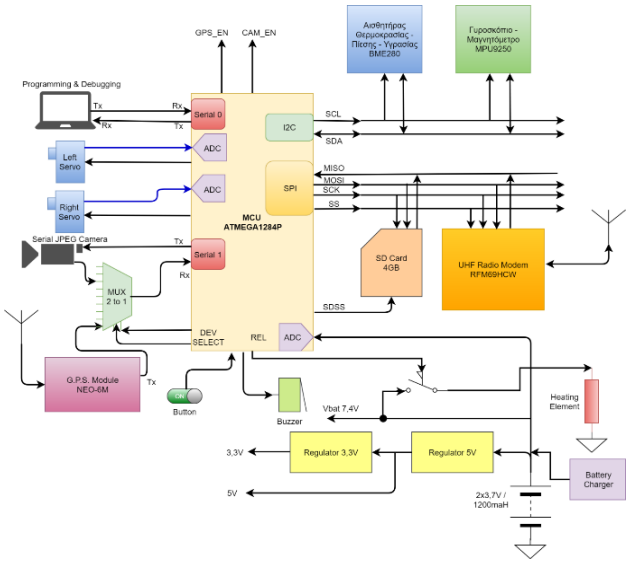

Fig. 10: The block diagram of the cansat.

II.II.I Batteries and mains switch

We used two 1200mAh Li-Po batteries each. This energy capacity is sufficient to provide up to 4 hours of autonomy. The batteries are connected in series so that the supply voltage of the cansat is 7.4V. The battery power is interrupted by a dip switch*. Because the dip switch is double, the other part is used to change the battery connection during charging. By switching, the batteries are parallelly connected. Also, next to the dip switch, there is the battery charger connector.

II.II.II Voltage Regulator

Because, as shown in Table 1, some components function at 5V and some others at 3.3V, we put two regulators on the microcontroller board. Initially the voltage is lowered to 5V with a 7805 and then regulated at 3.3V with a LD33CV.

Component	Max Current	Operating Voltage
ATMEGA 1284P	30 mA	5V
G.P.S.	70 mA	5V
Camera	75 mA	5V
Servos	2x210 mA	5V
BME280	1 mA	3,3V
MPU9250	5 mA	3,3V
SD Card	100 mA	3,3V
Radio Modem	135 mA	3,3V
Thermal element	450 mA	7,4V

Table 1: Operating voltage and current of the cansat components.

II.II.III Microcontroller

This is a hand-made arduino board with ATMEL Microcontroller Unit (M.C.U.) ATMEGA 1280P (ref. 2). We made this choice because the arduino micro did not fit all the code in Flash Rom and the Mega was too large for the dimensions of CanSat. Using this microcontroller, the code in its final form occupies 41% of Flash Rom and 22% of RAM.

Because the logic operating levels of the microcontroller are 0-5V, we used level shifters for the devices that use 0-3.3V .

II.II.IV Micro SD card

Secure Digital (SD) memory is connected to the MCU via the Serial Peripheral Interface (SPI) bus. Here, all measurements from the sensors and the position data of G.P.S. are stored in a file. Also, separate files with photos (640x480) and thumbnails (160x120) are saved.

II.II.V Microcamera JPEG

It is a color serial camera based on the VC706 chip. It communicates with the MCU via the 2nd Universal Synchronous Asynchronous Receiver

* This switch malfunctioned during launching, resulting in an interruption of power supply 162 meters prior to landing.

Transmitter (USART) which it shares with the GPS via a 2-in-1 Diode Resistor Logic (DRL) multiplexer, made with discrete components. The MCU can change various camera parameters such as resolution, compression ratio, communication speed, etc. The communication rate is set at 115200bps and transferring a 640x480 (~55Kbytes) image to the SD takes about 8-9sec, while other tasks such as reading, calculating, storing and transmitting measurements are performed, too. The MCU can turn off the camera via a PNP transistor so it does not consume power.

II.II.VI G.P.S. module

This is the NEO 6MV module that communicates with the MCU through the 2nd USART. To save space, we removed the active patch ceramic antenna and in its place we put a λ / 4 wire for the 1575 MHz frequency. This simple antenna gave a 30-40dB S / N ratio resulting in fast and stable positioning. The MCU can turn off GPS when energy levels drop.

II.II.VII BME 280 Sensor

It is a multi-sensor for atmospheric pressure, temperature and humidity. It is connected to the MCU via the Inter-Intergraded Circuit (IIC or I2C) bus.

II.II.VIII Gyroscopes & Magnetometer

It is a triple system of 9 degrees of freedom. It consists of a 3-axis gyroscope, a 3-axis accelerometer up to 16G and a 3-axis magnetometer. It is connected to the MCU via the I2C bus. We used the magnetometer for the compass of the GUI program and the measurement of the planet's magnetic field intensity. The gyroscopes and accelerometers were used to determine the position of the vehicle in 3D space and the presentation on the virtual horizon of the GUI.

II.II.IX Radio Modem

We used the RFM69HCW module in the UHF ISM 434Mhz band. It communicates with the MCU through the SPI channel. We used Gaussian Frequency Shift Keying (GFSK) with a transmission rate of 38,400 bps. The Packet Error Rate (P.E.R) was acceptable at a signal strength level of -87dbm*. The transmission of one 640x480 photography lasts about 50sec with a signal better than -80dbm. That is, the actual transfer rate is about 9-10 Kbps. When transmitting the image, transmission of the measurements is frozen.

II.II.X Micro servo motors

Wheel drive is provided by two MG90S micro servos with metal gears and a 360 degree rotation capability. Rotational speed and torque are satisfactory. The servos are powered by 5V and the MCU can adjust the number of revolutions. For obstacle detection, we measure the current intensity for each one through two analog inputs of the MCU. If it surpasses a level, then it manages to avoid the obstacle.

II.II.XI Thermal element

To melt the nylon thread, we use a common resistor of 15 ohms and a power of 1 / 4W. It is connected directly to the batteries, ie at a voltage of 7.4V. Following a command from MCU, a NPN transistor becomes conductive for 20-40sec until the thread is cut.

II.III Ground Station

In the diagram below (Fig. 11) all the components of the ground station and the interconnections between are shown. Here, an Arduino micro is more than enough. It is connected to the RFM69HCW radio modem via the SPI bus.

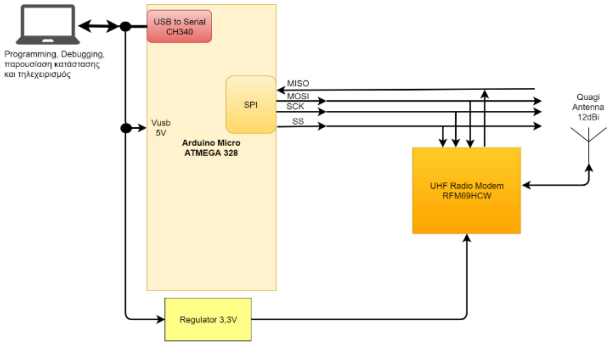

Fig. 11: The block diagram of the ground station.

* Even though we had LORA protocol modules, time pressure prevented us from using them. Since LORA is more advanced and also has F.E.C., we could have a reliable communication at -110 to -120 dbm signal level.

The arduino micro is powered by 5V from the USB port of the laptop. There is also a LD33CV regulator to supply the 3.3V radio modem with power.

II.III.I The directional antenna

To strengthen the CanSat low UHF signal, we use a Quagi 8-elements antenna. This antenna has a reflector and a Quad-type feeder and yagi-type directors. Its gain is about 12dbi and, owing to modifications, we achieved a SWR 1:1.1 to 1:1.2 ratio for 5MHz bandwidth.

During launching, when the antenna was pointing at the cansat, the signal level did not fall below -75dbm. At landing the signal went down to -80dbm but the cansat fell very close to the ground station and there were no obstacles in between.

II.IV The recovery system

We chose a cross-type parachute because of its ease of construction. Using a spreadsheet, we computed the dimensions of the parts using the formula (ref.3):

$$A = \frac{2mg}{\rho\, Cd\, V^2}$$

Where :
m = mass in Kg
g = acceleration of gravity 9,81 m/sec^2
ρ = density of air 1,22 Kg/m^3
Cd =drag coefficient 0,8
V=velocity

Fig. 12: The cross chute.

Making it, we used ripstop nylon fabric which is durable, waterproof and lightweight (Fig. 12). While descending, it successfully opened up and there were no instabilities and turbulence. The exact drop velocity was 7.68 m / sec.

III SOFTWARE & COMMUNICATION PROTOCOL

Both the CanSat and the ground station firmware are written in the Arduino wiring language. For development we used the Arduino IDE.

III.I CanSat

Constructionwise, the cansat software is structured and modular, so that it can be easily corrected and further developed. It is divided into many small files, with .h (header files) extension, each of which contains the part of code that pertains to each component. Some modifications have been made to the rfm69.h library, so that it performs well according to our requirements.

Functionwise, in a rough mode, the following are done:

- At startup, **once** with setup (), it prepares all components and checks if everything is ok, i.e. Power On Self Test (P.O.S.T.). If something malfunctions, it notifies us with a sound message and a change in the way the green L.E.D flashes.
- It controls **continuously** with polling, if there is a command from the ground station that needs to be decoded and executed.
- If it is a requirement to confirm a packet from the ground station then it responds accordingly.
- With using a timer interrupt, every n seconds and, regardless of whether it performs another task, it checks all sensors, makes calculations and conversions and transmits a packet to the ground station. If the ground station does not respond that it received the packet, it tries again 6 times.

III.II Ground station

Here things are much simpler. The code is much smaller and performs the following steps:

- It checks whether a packet has arrived from the satellite. If so, it determines if it is a measurements or a photo packet, sending it to the GUI through the serial port.
- If it is a request for packet confirmation it responds to the satellite.

- It controls the serial port buffer if a command is sent from the GUI. If yes, then it sends the command to the satellite. It waits for a confirmation and tries 6 times.

III.III Communication protocol

Although the RFM69 library can also handle binary data, we only send American Standard Code for Information Interchange (ASCII) character data through the telemetry channel, since it is easier to process them as a string.

The packets sent by the satellite to the ground station have a total length of up to 60 bytes, according to the following structure:

Packet no	Type	Payload

Values are always separated by the character ';'

Packet no is the serial number and it can reach up to 65535. Then it returns to 0.

The types are:

III.III.I The 'S:' type

Type S: is for sensor readings with a frequency of one packet per second*, e.g.

```
56;S:;T;P;H;B.V.;Roll;Pitch;Yaw;M.F.I.
```

Where:

56 = Packet No
S: = Type
T = Temperature
P = Pressure
H = Air humidity
B.V. = Battery voltage
Roll = Roll of vehicle
Pitch = Pitch of vehicle
Yaw = Yaw of vehicle
M.F.I. = Magnetic field intensity

III.III.II The 'G:' type

Type G: is for GPS data. These packets may be transmitted more slowly, e.g. one every four seconds.

The structure of a G packet is like in the example:

```
57;G:;T;D;LAT;N/S;LON;E/W;V/A
```

Where:

57 = Packet No
G: = Type
T = Time
D = Date
LAT = Latitude
N/S = North or South
LON = Longitude
E/W = East or West
V/A = 'A' for valid GPS data

III.III.III The 'Re:' type

The Re: type is CanSat response packets to the ground station. When a command is sent to CanSat, it responds with a Re: packet followed by the way it performed in order to show up in the GUI.

III.III.IV The photograph packets type '$'

The type '$' are packets with photo data and they have the format:

```
1258;$;D
```

Where:

1258 = Packet No
$ = Type
D=Image data in ASCII characters with a maximum length of 52 characters.

The images are initially stored in the micro SD memory, while the microcontroller performs all necessary functions, such as transmission of measurements and decoding of commands. When the ground station asks for an image, the program reads the binary image file and splits it into 39 bytes segments. Each section is encoded in Base64 code, where every three bytes one more is inserted (ref. 4). Thus, while we initially had 39/3 = 13 triplets, after encoding, there would be 13 quadrants, that is 13 * 4 = 52 characters. Base64 encoding produces ASCII characters.

The ground station forwards the image packets to the GUI program which performs the reverse process and stores the binary JPEG image on the

* A GUI command can vary the frequency. During launching it can be 4 packets / sec in order to obtain more detailed graphs.

hard disk. When storing is finished, it displays the image on the screen.

III.IV The GUI control program

The GUI program runs on the ground station laptop. It is written in Processing Language and is modular and structured so that it can be easily corrected and furthermore developed. The 'S', 'G', 'Re' and '$' packets are sent from the arduino micro of the ground station to the PC via the USB port. The program classifies, decodes them and displays the results.

At the same time, the user, by pressing buttons on the computer screen, can send commands to CanSat such as vehicle navigation commands, release from the lander or change of some satellite parameters Fig. 13.

Fig. 13: Snapshot of the GUI program on the launch day. Sensor measurements are shown on the top left, the virtual horizons and the compass on the bottom left, the photo and geographic position on the map, and the control and navigation controls on the bottom right.

IV SCIENTIFIC RESULTS

IV.I Primary mission

During the main mission we got all the required data. The most important charts are as follows :

IV.I.I Pressure and altitude

Cansat sends air pressure measurements. Altitude calculations are made in real-time by the GUI program and calculated on the spreadsheet.

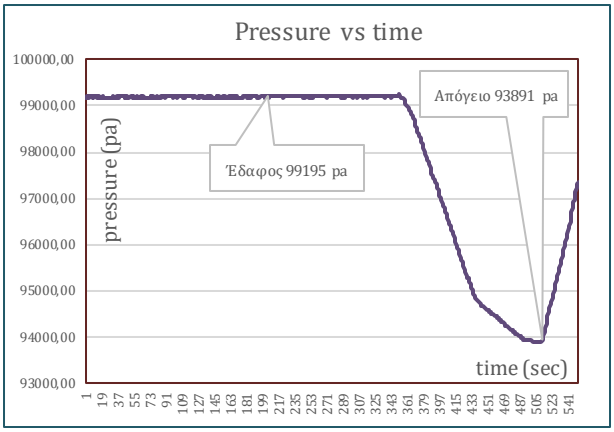

Fig. 14: Air pressure variation vs time.

At first, we calculated the altitude based upon atmospheric pressure data, using the formula (ref. 5):

$$alt = -45846{,}2\left[\left(\frac{press}{pressOffs}\right)^{0{,}190263} - 1\right]$$

Where 'press' is the current value of the pressure in pascal and 'pressOffs' is the value of the pressure at the surface of the sea. Since we fed the 'pressOffs' value shortly before the launch, the altitude we calculated is relative to the ground rather than the sea level.

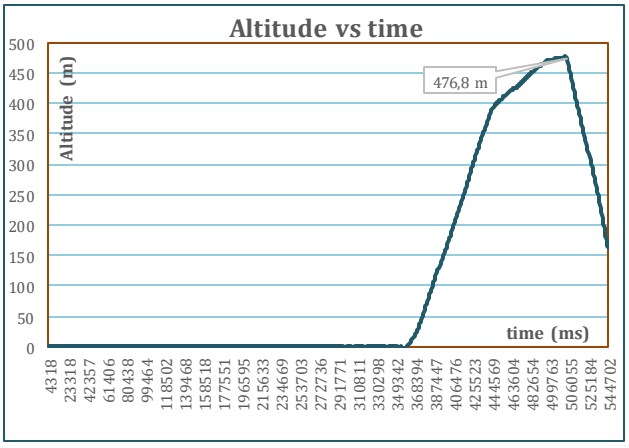

Fig. 15: Relative altitude variation vs time.

When the cansat is activated, it gets the first value as the initial 'pressOffs'. Unfortunately, due to the failure of the main switch, the cansat stopped operating at 162 meters before landing. Consequently, we have no data.

IV.I.II Air temperature

As shown in the graph of Fig. 16, the temperature inside the cansat rises slightly as the cansat is on and it is on the ground. As soon as the cansat rises, the temperature drops and at the apogee it drops to 27.5 ° C. When the cansat descends, the temperature goes up.

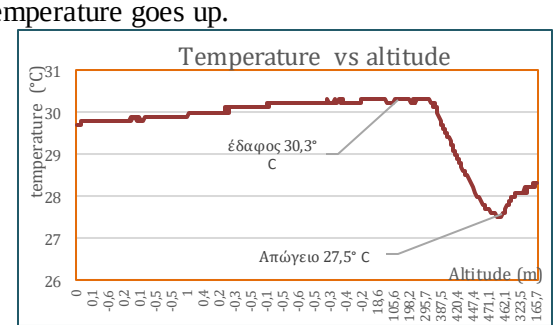

Fig. 16: Temperature variation vs altitude.

IV.I.III G.P.S. Data

The position data was transmitted continuously from the cansat at a frequency of one sample per three sec. The GUI has converted the data from the NMEA sentence to decimal degrees, in order to present the position on Google Maps.

Fig. 17: Display of real time geographical position from GUI.

IV.II Secondary mission

Also, during the secondary mission we got all the data we had set as our goal. Here are the most important charts:

IV.II.I Air humidity

The graph below shows the change in humidity as relates to altitude. The triple sensor BME280 sends air humidity values.

As shown in the graph, air humidity decreases as height increases. When the cansat descends, humidity tends to increase.

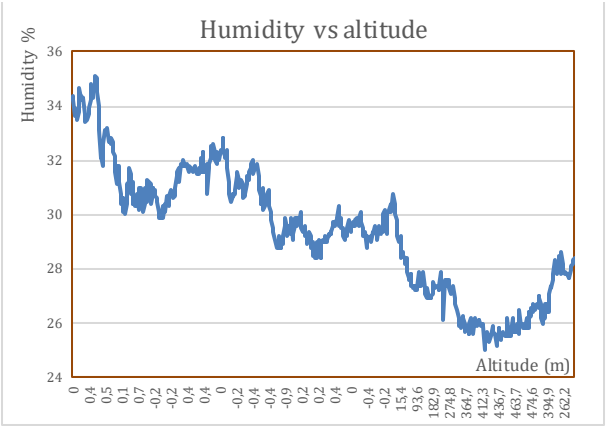

Fig. 18: Humidity variation vs altitude.

IV.II.II Magnetic field intensity

As it was previously mentioned, the cansat has a three-axis magnetometer. We calculate the composite intensity of the planet's magnetic field using the formula:

$$E = \sqrt{\varepsilon_x{}^2 + \varepsilon_y{}^2 + \varepsilon_z{}^2}$$

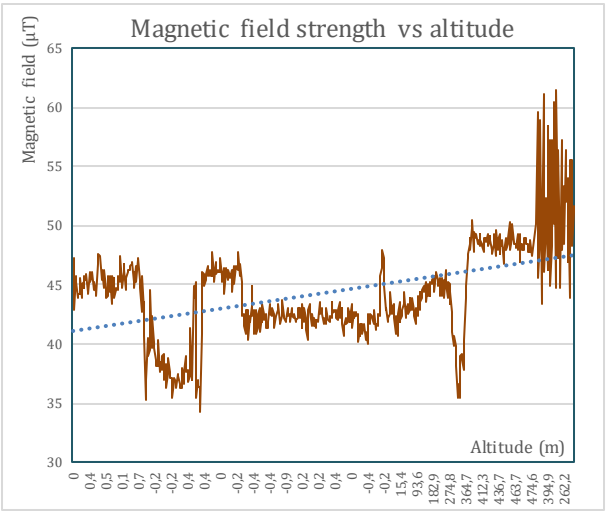

Fig. 19: Magnetic field intensity variation vs altitude.

IV.II.III Photographs

The cansat successfully managed to store and transmit color photos through the telemetry channel. Upon order from the GUI, cansat displayed a low resolution snapshot, so the operator can see what is ahead of the vehicle. If requested, it can store the

image with higher resolution (640x480) on the computer's hard disk.

Fig. 20: Low resolution snapshot from launching area.

V DISCUSSION

In both missions, the data we received was in line with the original plan and gave reasonable scientific conclusions.

V.I Primary mission data

According to Fig. 15, we see that our satellite reached the apogee 2.4 minutes after takeoff. The ascending velocity is stable up to 400m and we calculate it at 5.3 m / sec. Descending velocity is 7.63 m / sec. For the calculation we used the formula:

$$v = \frac{S}{t}$$

From Fig. 16 we can conclude that as long as the cansat is stationary and in operation, the temperature rises due to the heat of the electronic components. During launching, it is testified that at high altitude we have a temperature drop.

The GPS data was very accurate with a deviation of 1-3 meters. Our cansat did not send height data from the GPS because we thought it was not that accurate.

V.II Secondary mission data

According to Fig. 18, we notice a relative humidity drop as the altitude rises. The day of the launch was a warm and dry day. Perhaps, if there had been cloudiness and the satellite was entering a cloud, then we would have recorded an increase in relative humidity.

The gyroscopes and accelerometers were used to navigate the vehicle. Although we had data, we did not process them to draw any scientific conclusions.

We mainly used the magnetometer as a compass for navigating the vehicle. According to Fig. 19, we notice that the mean value of the magnetic field strength is 45 μT, a normal value if we take into account that there are several metal components close to the sensor. We also notice large fluctuations in the fall of the cansat due to the chute turbulence.

The photos were stored in the SD card and transmitted from the cansat to the ground station.

VI CONCLUSIONS

Despite the fact that, on the day of the launch, we had two failures, the whole project was successful because the original plan was achieved, in particular in the following goals:

1. Telemetry system.
2. All sensors have provided reliable data, which led us to scientific conclusions.
3. The GPS system has provided a reliable and stable position.
4. Data and photos were stored in the SD memory.
5. The crosschute.
6. Release from the lander and expansion of wheels.
7. Navigation and taking photos from the ground.

The failures were as follows:

1. At the last minute, we decided to change the ground station laptop, causing the antivirus program to hang the GUI, while the satellite was in the air. As a result, we could not get all the telemetry data on the laptop and we could not command the cansat to take photos from the air in time.
2. Due to the defective main switch, the cansat stopped operating before landing. So, we lost some of the data and the rest of the mission continued after our own intervention. In the end, the data we have processed is what we stored on the SD card.

We believe that if we had done more tests with drops from an altitude, we would have avoided these failures.

However, the benefit from this project was very important, because we learned a lot of things in various subjects, we had the experience of building and planning such a great project and, on top of everything, the value of teamwork.

Finally, the experience of the actual competition was unique.

VII REFERENCES

1. http://www.ucs.louisiana.edu/~jev9637/research/ARLISS.html
2. http://ww1.microchip.com/downloads/en/DeviceDoc/doc8059.pdf
3. http://www.rocketmime.com/rockets/descent.html
4. https://en.wikipedia.org/wiki/Base64
5. https://ccollins.wordpress.com/2016/03/07/arduino-altimeter/