7. DEVELOPMENT OF APPLICATIONS

For this study two applications were developed up to a functional prototype stage. Both applications were of aeronautical nature: the first one is a module for helium balloons that takes temperature samples from inside the module, the atmosphere and IR temperature scans from the sky above the balloon and the second one is a module intended to be installed in drones which enables them to make a temperature map of a terrain.

7.1. HELIUM WEATHER BALLOON TEMPERATURE MODULE

The first application developed was a module designed to be installed in a weather helium balloon which would take different temperature measurements that, coupled with the altitude information, provide useful information for weather prediction and atmosphere monitoring.

The first of the measurements was the outside atmospheric temperature, taken with a contact sensor. This information is usually used in order to know the temperature gradients in the atmosphere at different heights in order to predict the forecast [25]. It was also interesting as the data can be easily compared with the ISA atmosphere model. The second was an IR temperature measurement of the sky above the balloon. This temperature can be useful for the study of the greenhouse effect [31], sky transparency and cloud detection [32]. For this study it was used as a test to the measures at high altitude, where the temperature measured should be near the absolute zero as the IR sensor would be pointing to the space with minimal interference of the atmosphere. The last temperature was the temperature inside the module itself, useful in order to improve the isolation and heating systems of the module, as it will be explained afterwards.

7.1.2. THE NESLAB PROJECT

The development of this application would not have been possible without the aid of a student-project based at ETSEIAT University named Neslab. This student-project develops experiments and launches them with helium balloons up to high altitudes (around 30Km height). The structures, tracking system and launching system are also self-developed.

The project offered to launch this application on one of their launches held on the 22nd of December of 2014. This made possible the test of the module in a real environment

of operation, where the data showed in section 7.1.7. was obtained. In order to keep the weight of the whole payload low (the module developed in this section was launched along with other experiments) the electrical power used by the module developed in this work as well as the altitude information would be supplied by the modules developed by the Neslab team.

7.1.3. ELECTRICAL POWER AND INSULATION

Even if the power system was developed by the Neslab team, it is worth to explain it as it is a critical part of the whole launch. One of the main problems when carrying electrical devices to high altitudes is its batteries. Even if the equipment has a low electrical requirement, batteries capacity hugely diminish when exposed to cold environments. Some sensors can also malfunction due to cold conditions.

In minimize this problem, active and passive measures were taken. A system of electrical resistances was installed which would heat up the interior of the structure were the batteries and electronics were held. Also, the structure itself was made of foam which diminished the heat loss due to conduction and the interior and exterior surfaces were covered with thermal blankets to minimize radiation heat transfer.

The Arduino can be electrically supplied through its input power pin, which accepts a voltage between 6 and 20V (although it is recommended to use a 7 to 12 voltage), and internally it converts this voltage to the 5V used for all the Arduino functionalities. However, it can also be fed through its 5V output but then the input voltage must be 5V as the Arduino does not have any voltage regulator if that port is used as an input. For this project, this second method was used because even though it is more delicate, it does not waste energy transforming the voltage to 5V which means less electrical consumption. Due to the cold problem, the voltage delivered by the batteries would not be a stable value but fluctuate, for this reason a voltage controller was installed in order to ensure a 5V stable power bus.

For power redundancy, two sets of batteries were installed. Both would be able to supply the 5V needed to power the Arduino even at cold conditions. One of the batteries would rest unused and switch on in the event of a failure of the other one. The system was tested in a cooler in order to corroborate that the batteries would provide the electrical power required.

7.1.4. DATA LOGGING

It was set as a requirement for the applications that they had to be standalone. In order to carry out this requirement, the devices needed to have data logging capabilities. For this application this requirement is evident as weight and electrical consumption are a critical element of the module design which makes impossible to include a whole computer for that matters. One way to cover up this necessity is adding to the system an SD logging device, which gives the capacity to the Arduino to save the sensors data in one or more text files into an SD or micro SD card.

7.1.4.1. Wiring for data logging with SD cards

Most SD cards use SPI communication protocol, which means that they need to be plugged to the Arduino pins which support this communication protocol (digital pins 10 to 13). However, the connection between the Arduino and the SD card logger cannot be direct due to the fact that SD card logger pins work on a 3,3V logic level while the Arduino's work on a 5V, which makes the use of a level shifter compulsory. Adafruit's BSS138 4-channel I2C-safe Bi-directional Logic level converter was used for this purpose, as it already have 4 independent level shifting channels. The wiring is shown in Figure 1 and goes as follows:

- Arduino ports 10, 11, 13 and 12 are plugged to the B1, B2, B3 and B4 pins of the level shifter respectively.
- A1, A2, A3 and A4 ports from the level shifter are connected to the SS, MOSI, SCK and MISO ports of the SD card data logger respectively.
- The 5V and 3,3V supply and the GND pins form the Arduino are plugged to the HV, LV and GND ports of the level sifter respectively.

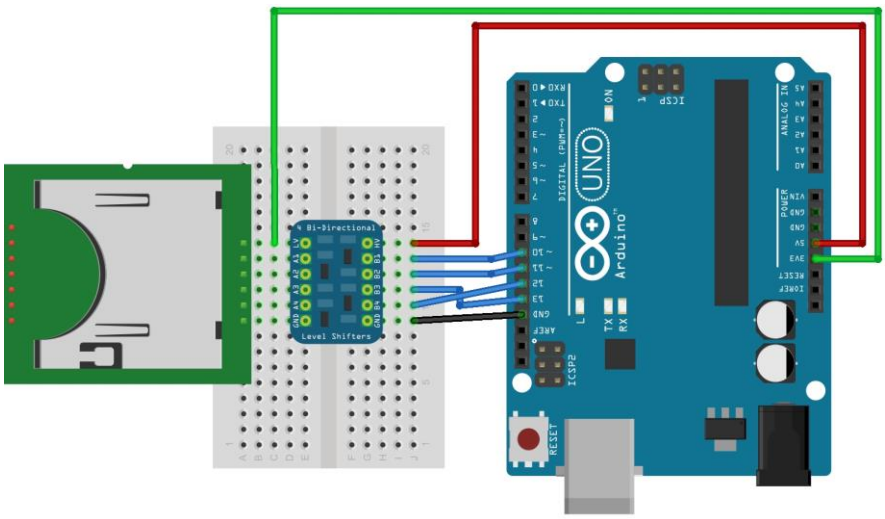

Figure 1: Wiring diagram of the SD card logger with the Arduino.

7.1.4.2. Software for data logging with SD cards

In this section the main instructions needed for data logging to an SD card which works with SPI communication will be explained. Most of these instructions can be seen in the Arduino SDlogging program example.

7.1.4.2.1. Libraries

- **SPI.h:** This library includes helpful instructions for SPI communication in general.
- **SD.h:** This library has all the instructions needed to communicate with an SD card device.

7.1.4.2.2.Global variables

- **Int *chipselect*:** An int variable named *chipselect* is created and assigned to 10. This variable will later be used to define which port of the Arduino is the one that is in charge to select which SPI device is communicating to (this is, the SS port).
- **File *logFile*:** A variable of type file is created, which will be used in order to create the files inside the SD card.

7.1.4.2.3. Setup

In the Setup, the serial communication is started, the SD card initialized and the file where the data is going to be saved created.

- **Serial.begin(9600):** This command initializes the serial port for terminal-communication with the PC, useful for debugging.
- **pinMode(*chipselect*, OUTPUT):** this command will set the *chipselect* port as an output in order to send instructions to the SPI devices connected about which one is it talking to. In this case, as there is only one SPI device this feature will not be used, but it is important in case of holding more than one SPI device.
- **SD.begin(*chipselect*):**With this instruction the SD card is initialized. This function returns a Boolean, true if the SD card could be initialized and false if it did not, so it is usually set as an "if" condition in order to send a failure message through the terminal port in the case of not being able to initialize it.

If the program is wanted to create a new file each time the program executes instead of always writing over the same file, the following process can be used:

- A **char** type variable named *filename* is created.
- With the **strcpy(*filename,string*)** a *string* is assigned to this variable. This string will be the root of the file name. As an example, *string* could be *"log00.txt"*.

- With a ‘for’ loop, the two zeros of the filename will be sequentially changed by adding 1 unit. Inside this loop there is an “if” structure with the instruction **!SD.exists(*filename*)** as condition. If there already is a file saved in the SD card with the name proposed, the “for” loop will continue until a non-existent file name is found.
- Finally, the file is created and opened for writing with the ***logfile*=SD.open(*filename*,FILE_WRITE)** instruction and can be referenced directly with the *logfile* variable.

7.1.4.2.4. Loop

In the loop, data can be written in the file by using the ***logfile*.print()** instruction. Also, the instruction ***logfile.*flush()** is used. This instruction waits for the transmission of outgoing serial data to complete and is used in order to prevent data loss.

7.1.5. MATERIAL

The Arduino used for this module was not the Arduino UNO but the Arduino Nano. The reason behind this decision is that this version of the Arduino is much smaller and thus lighter, making it more suitable for an application where weight and size are critical; and its pins can be soldered, which diminishes the probability of accidental disconnection of lines due to movement of the structure. It also runs on the same microprocessor so all the programs are compatible, as well as the pin’s configuration. The Nano used was not an original Arduino, but a clone developed by Funduino, which was lent by the Neslab team, but can be bought for around 9€, depending of the dealer.

The sensors used were the two 100KΩ thermistors and the Melexis 90614-ACF IR sensor. One of the thermistors would be placed inside the module in order to track the inside temperature of the module, one would be outside the module in order to track the atmospheric temperature and the Melexis IR sensor would be outside the module too but pointing 45º down with respect to the XY plane using the body-axes system. This last sensor was supposed to point directly vertical up to the sky in order to track the temperature of the sky above, but if that was done the balloon would be inside its field of view and would interfere with the readings. For this reason it was given an angle with enough margins so that if the module shacked there was no chance that the balloon would end up inside the field of view of the sensor.

The SD card data logger was the LC Studio SD card data logger, obtained for 1,40€. The Level shifter used, as mentioned before, is the BSS138 4-channel I2C-safe Bi-directional Logic level converter.

The overall cost of the whole module was of 36,54€. The price breakdown can be found at Table 1.

Item	Unitary Cost	Number of units	Total cost
Melexis 90614-ACF IR sensor	17,45€	1	17,45€
100KΩ thermistors	1,20€	2	2,40€
Studio SD card data logger	1,40€	1	1,40€
Adafruit BSS138 4-channel Logic level converter	3,30€	1	3,30€
Funduino Nano	8,99€	1	8,99€
Other (wires, drilled plate, resistances, etc)	-	-	3€
Total			36,54€

Table 1: Cost breakdown of the weather helium balloon module.

7.1.6. WIRING

The wiring of the sensors would be a superposition of the previously presented wirings at section **¡Error! No se encuentra el origen de la referencia.** for the thermistors and the Melexis IR sensor. The SD data logger would be wired as explained in section 7.1.4.1. Wiring for data logging with SD cards. The wiring of all the system can be seen in Figure 2.

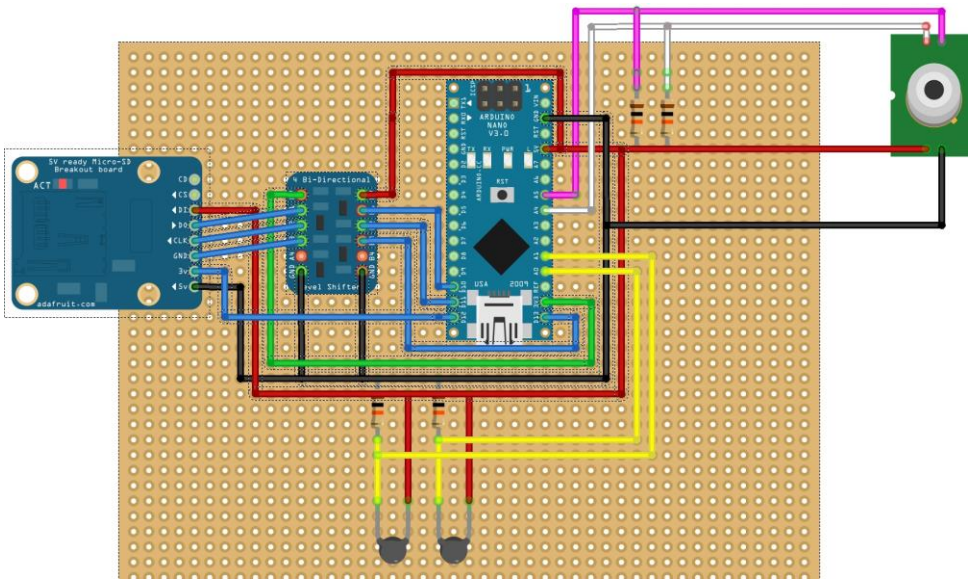

Figure 2: Wiring diagram of the weather balloon module.

7.1.7. SOFTWARE

For this application only the Arduino code will be explained, as the post-process code was as simple as importing the data to matLab and plotting it. However, the matLab code can be found at the annex section 1.3.2. MATLAB CODE (POST-PROCESS).

The code for Arduino is a combination of the codes for the thermistors and the Melexis IR sensor which are explained at section **¡Error! No se encuentra el origen de la referencia.**, adding the code necessary for data logging into an SD card explained previously at section 7.1.4.2. Software for data logging with SD cards. The full code can be found at the annex section 1.3.1. ARDUINO CODE.

The program works as follows:

- The libraries for communication with the Melexis sensor and the SD logger are included and the global variables bond to the thermistors and the log file defined.
- In the setup, the Melexis sensor and the SD card are initialized and a file created into the latter with a unique file name.
- In the loop, the sensors data is read by the Arduino using the ***mlx*.readObjectTemp()** and the **analogRead(*thermistor*)** instructions. Then, it is saved to the SD card by using the ***logfile*.print()** instruction. A **flush()** is added in order to ensure that the data have been saved and a **delay(1000)** so that there is a spacing of 1 second between every measurement.

7.1.7. TEST IN A REAL ENVIRONMENT

The module was installed in the Neslab launch on the 22nd of December of 2014. The helium balloon carried two payloads, one containing different modules developed by the Neslab team and the one developed in this project and the other developed by old veteran team members from Neslab.

7.1.7.1. Launch site and climatology

The module was launched from Bell-lloc d'Urgell, Catalonia, Spain at 8:43am (367m height from sea level) and recovered the same day at 11:37am near Coromines, Catalonia, Spain (see Figure 3). It reached 29,42Km altitude.

The climatology for that day on morning was good but with a thick fog at an approximate height of 1000m (**¡Error! No se encuentra el origen de la referencia.**), due to an inverted gradient of the atmospheric temperature. This fact will be visible in the data collected, as it will be seen later. The climate predictions for that day permitted to estimate the flight path and the landing site, which only differed from the actual landing site by 7km.

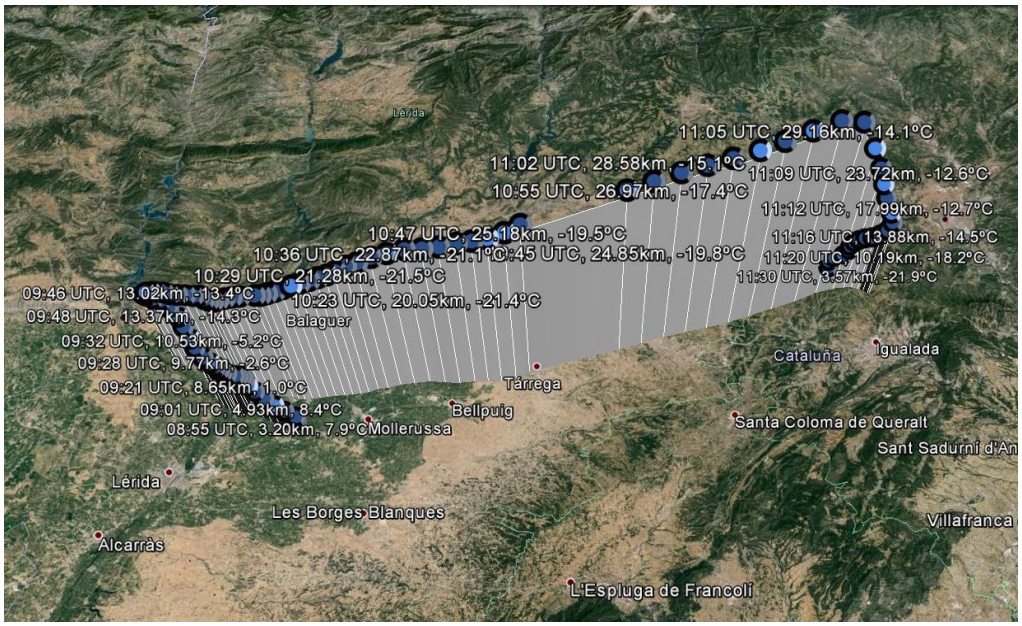

Figure 3: Image of the 22nd of December of 2014 Neslab launch flight path.

Figure 4: Photograph obtained from an on-board camera of the Neslab balloon, when it was about to enter the fog. The payload that is showed is the one developed by the old veteran team members from Neslab.

7.1.7.2. Results

Due to the fact that the altitude data had a higher monitoring time, at the end of the experiment there were more temperature data points than altitude's, so altitude points were created between the actual altitude data points assuming a linear increase in order to match the number of temperature data points.

The data obtained for the three temperature sensors against altitude for the ascending phase can be seen in Figure 5.

. In the same graph the ISA (international standard atmosphere) temperature has been plotted for comparison with the measured atmospheric temperature. The data from the thermistors and the IR Melexis sensor has been calibrated. The descent data has been discarded as it does not add useful information for the analysis. However, all the data of the experiment can be found at the annex section 3.1. HELIUM WEATHER BALLOON MODULE.

The data obtained by the thermistor placed outside in order to track the atmosphere temperature is shown in dark blue. The temperature at ground level (367m from sea level) was about 1ºC. The temperature then increased up to 17,82ºC at 1651 meters height. Afterwards it would decrease until hitting -44,43ºC at 18,2km, where it would start heating until the end of the climbing at 29,42km, where it reached -15,38ºC.

The Melexis IR sensor's data (green) is constantly diminishing but at different rates depending of the altitude. Its measurements start at -27,23ºC and diminish slowly until -33,09ºC are hit at 2621m height (2254m relative to the ground level). After this the measurement variation increases, reaching -74,33ºC at 8,03km height. From this point up, the temperature slowly starts to tend to a temperature asymptote, being the closer value to the limit temperature the last data point taken at 29,42km, -119,50ºC. It is worth noting that in this last stage there is increasing measurement noise as the height increases, although it looks like it stabilizes at 17km where the noise is about ±5,5ºC with some high temperature peaks.

The data obtained by the thermistor placed inside the module is shown in red. From launch to 6,32km height the temperature slightly rises from 17,02ºC to 19,36ºC. From that point until 10,52km the temperature remains fairly stable and then starts to drop until 6,20ºC are reached at 29,42Km.

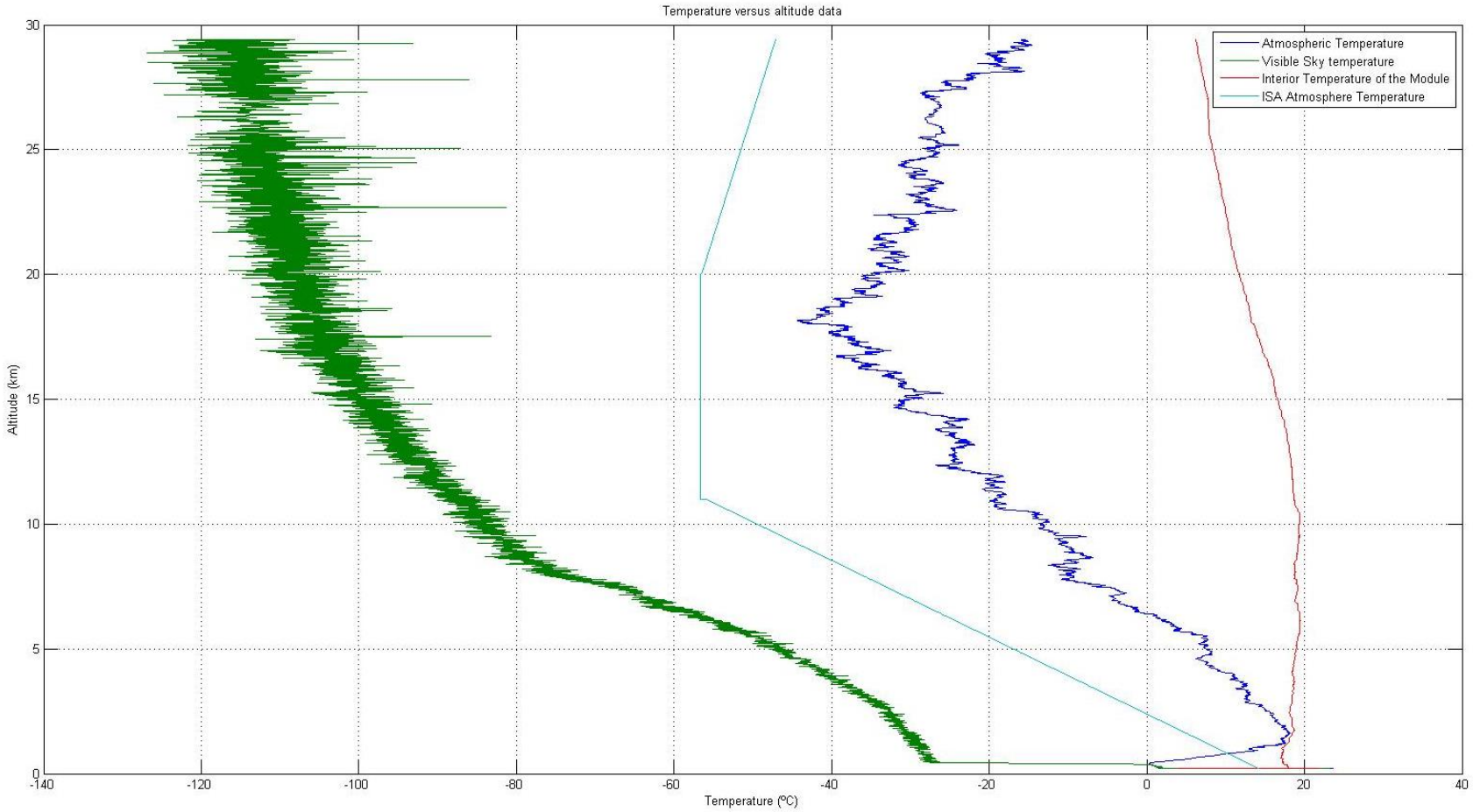

Figure 5: Sensors data and ISA temperature profile versus altitude.

7.1.7.3. Analysis of the results

As it was explained in section 7.1.7.1. Launch site and climatology that day there was an inverse thermal gradient. This fact is reflected in the initial increase of temperature of the outside thermistor. Afterwards the temperature decreases as expected. It can be seen that the thermal gradient is not as shallow as the one for the troposphere proposed by the ISA temperature model, but they are fairly even. However, when the end of the troposphere is reached, the temperature keeps decreasing instead of remaining constant as the ISA temperature model proposes. At 18,2km the temperature starts rising again due to the fact that the layer with maximum ozone concentration is reached. This behavior is modeled by the ISA at 20km and it can be observed that the gradients are fairly even. The higher temperatures of the actual data versus the model data may be due to the low atmospheric pressure of that day.

The Melexis IR sensor measurements firstly decrease at a slow rate. This is due to the fact that there was thick fog so the IR sensor measurement is actually the fog's temperature. As it enters the fog and the sensor has less thickness of it above, the rate at which the temperature decreases increases. When it leaves the clouds, as height is gained, less atmosphere and thus humidity interferes with the IR measurements, slowly reducing the gradient of the measured temperature, tending to stabilize. Although not enough height was gained in order to completely stabilize the measurement, its tendency to a certain value is visible.

Finally, the increase of noise at 8km can be produced for several reasons:

- At 8km height the temperature measured reaches -70ºC, which is, according to the manufacturer, the minimum temperature it can measure. For this minimum temperature, the error of the sensor is ±4ºC [25], which is approximately the noise it is getting.
- Due to the fact that the payload was attached to a single rope, it could spin. This could lead to moments when the IR sensor was pointing towards the sun and others were it was pointing away from it. This fact would increase the noise and could be the explanation to the hot peaks that can be seen especially over 17Km.

The interior thermistor shows a fairly constant value. The warming system as well as the insulation does a great job. However, an interior-temperature control could be designed in order to minimize the electrical consumption, as the temperature does not need to be held so high and it even increases for the first section of the flight. On the other hand, at the end of it, temperature decreases significantly (although not being critical). A simple temperature control could manage the temperature more efficiently, keeping the electronics at its optimal temperature of operation reducing electrical consumption.

10. BIBLIOGRAPHY

[1] T.P.Wang, "Thermocouple Materials," *ASM - The Materials Information Society,* pp. k88 - k107, 1990.

[2] Maxim Integrated, "Implementing Cold-Junction Compensation in Thermocouple Applications," 26 April 2007. [Online]. Available: http://www.maximintegrated.com/en/app-notes/index.mvp/id/4026. [Accessed 03 November 2014].

[3] National Instruments, "Cómo Realizar una Medición con Termopares," 19 August 2013. [Online]. Available: http://www.ni.com/white-paper/7108/es/. [Accessed 02 November 2014].

[4] A. Tong, "Improving The Accuarcy of Temperature Measurements," 2001. [Online]. Available: http://www.picotech.com/applications/temperature.html. [Accessed 2014 November 03].

[5] National Instruments, "Guía para Realizar Medidas de Temperatura con RTDs," 19 August 2013. [Online]. Available: http://www.ni.com/white-paper/7115/es/. [Accessed 03 Noveber 2014].

[6] Agilent Technologies, "Practical Temperature Measurements," 26 January 2012. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5965-7822E.pdf. [Accessed 03 November 2014].

[7] M. Massoud, Engineering Thermofluids: Thermodynamics, Fluid Mechanics, and Heat Transfer, Berlin, Germany: Springer, 2005.

[8] Scigiene Corporation, "Infrared Thermometers," [Online]. Available: http://www.scigiene.com/pdfs/Infrared%20Thermometers.pdf. [Accessed 03 November 2014].

[9] OMEGA, "Introduction to Infrared Thermometer," [Online]. Available: http://www.omega.com/prodinfo/infraredthermometer.html. [Accessed 03 November 2014].

[10] OMEGA, "Digital Thermometer HH11B," [Online]. Available: http://www.omega.com/pptst/HH11B.html. [Accessed 02 December 2014].

[11] OMEGA, "RDXL-SD Series Portable Thermometer/Data Loggers with SD Card and Thermocouple Input," [Online]. Available: http://www.omega.com/pptst/RDXL-SD_SERIES.html. [Accessed 02 December 2014].

[12] OMEGA, "Compact Portale Data Logger," [Online]. Available: http://www.omega.com/pptst/RDXL120.html. [Accessed 02 December 2014].

[13] AllQA, "AllQA Infrared Thermometers," [Online]. Available:

http://www.allqa.com/infrared.html. [Accessed 06 December 2014].

[14] Fluke, "Fluke VT04 Visual IR Thermometer," [Online]. Available: http://www.fluke.com/fluke/m3en/thermometers/infrared-thermometers/fluke-vt04.htm?PID=77085. [Accessed 06 December 2014].

[15] National Instruments, "Dispositivo de Medidas de termopares NI USB-TC01," [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/es/nid/208177. [Accessed 07 December 2014].

[16] National Instruments, "Sistema de Medidas NI 9215," [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/es/nid/209871. [Accessed 07 December 2014].

[17] Arduino, "Arduino UNO," [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardUno. [Accessed 7 December 2014].

[18] B. Earl, "Learn Adafruit: Arduino Memories," Adafruit, [Online]. Available: https://learn.adafruit.com/memories-of-an-arduino/arduino-memories. [Accessed 07 December 2014].

[19] Arduino, "Arduino Software," [Online]. Available: http://arduino.cc/en/Main/Software. [Accessed 07 December 2014].

[20] Adafruit, "Thermocouple Type-K Glass Braid Insulated," Adafruit, [Online]. Available: http://www.adafruit.com/products/270. [Accessed 08 December 2014].

[21] Adafruit, "Adafruit 1-Wire Thermocouple Amplifier - MAX31850K," Adafruit, [Online]. Available: https://learn.adafruit.com/adafruit-1-wire-thermocouple-amplifier-max31850k/overview. [Accessed 08 December 2014].

[22] Adafruit, "4-channel I2C-safe Bi-directional Logic Level Converter - BSS138," Adafruit, [Online]. Available: http://www.adafruit.com/products/757. [Accessed 08 December 2014].

[23] Maxim Integrated, "DS18B20 Programmable Resolution 1-Wire Digital thermometer Datasheet," Maxim Integrated, [Online]. Available: http://datasheets.maximintegrated.com/en/ds/DS18S20.pdf. [Accessed 12 December 2014].

[24] Melexis, "MLX90614 family Single and Dual Zone Infra Red thermometer in TO-39 datasheet," Melexis, [Online]. Available: http://www.adafruit.com/datasheets/MLX90614.pdf. [Accessed 08 december 2014].

[25] Arduino, "Arduino playground: Reading a thermistor," Arduino, [Online]. Available: http://playground.arduino.cc/ComponentLib/Thermistor2. [Accessed 11 December 2014].

[26] Adafruit, "Using Melexis MLX90614 Non-Contact Sensors," Adafruit, [Online]. Available: https://learn.adafruit.com/using-melexis-mlx90614-non-contact-sensors. [Accessed 12 December 2014].

[27] Parr Instruments, "Plain Jacket Oxygen Bomb Calorimeter Instruction Manual," [Online]. Available: http://aui.ma/personal/~S.ElHajjaji/Labmanual/MSDS/InstructionManuals/BombCalorimeterManual2.pdf. [Accessed 14 12 2014].

[28] haake, "Haake Circulators Instruction Manual," [Online]. Available: http://www.geminibv.nl/labware/haake-q-koelwaterbad/haake-f3-series-circulators-manual-eng.pdf.

[29] ReRap, "PCB Heatbed," [Online]. Available: http://reprap.org/wiki/PCB_Heatbed. [Accessed 30 12 2014].

[30] J. Tyndall, Heat Considered as a Mode of Motion, London: Longmans, green and Co., 1868.

[31] Kitt Peak National Observatory, "Building a Thermoelectric Sensor to Measure IR Sky Transparency for Cloud, Cirrus and Dust Detection, at Night," [Online]. Available: http://www.noao.edu/staff/gillespie/projects/cloud-detector.html. [Accessed 02 01 2015].

[32] Adafruit, "Adafruit ultimate GPS logger shield," [Online]. Available: https://learn.adafruit.com/adafruit-ultimate-gps-logger-shield. [Accessed 02 01 2015].

[33] Science Miseim UK, "Warming world: Satellites and weather balloons," [Online]. Available: http://www.sciencemuseum.org.uk/climatechanging/climatescienceinfozone/exploringwhatmighthappen/2point1/2point1point2.aspx. [Accessed 02 01 2015].