



The aim of this project is to design and build a GPS tracker, a device that should be useful for the expert user in order to try new modules and getting them to work very quickly as well as for the new user that wants to get involved into electronics. This project differentiates from the others available at the internet due to the inclusion of two very cheap photographic sensors that are directly interfaced to the arduino board itself, thus cheapening the cost of the board a lot.



**Luis Sanchez Velasco** is a telecommunications engineer from Granada, Spain. He finished his Bachelor's Studies in 2017 at the University of Granada.

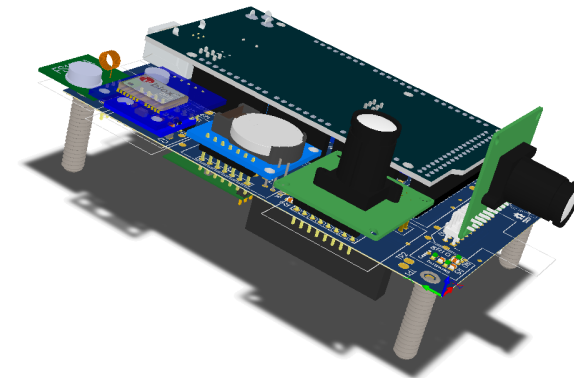


**Andrés María Roldán Aranda** is the academic head of the present project, and the student's tutor. He is professor in Department of Electronics and Computer Technology at University of Granada

## Design of an embedded camera with an AX.25 transmitter

Luis Sánchez Velasco

# UNIVERSITY OF GRANADA Bachelor Degree in Telecommunications Technology Engineering



Bachelor Thesis

## Design of an embedded camera with an AX.25 transmitter

Luis Sánchez Velasco

Academic year 2016/2017

Tutor: Andrés María Roldán Aranda



# UNIVERSIDAD DE GRANADA

GRADO EN INGENIERÍA DE  
TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

*“Design of an embedded camera with an  
AX.25 transmitter ”*

CURSO: 2016/2017

Luis Sánchez Velasco





# UNIVERSIDAD DE GRANADA

GRADO EN INGENIERÍA DE TELECOMUNICACIÓN

*“Design of an embedded camera with an AX.25  
transmitter ”*

REALIZADO POR:

**Luis Sánchez Velasco**

DIRIGIDO POR:

**Andrés María Roldán Aranda**

DEPARTAMENTO:

**Electrónica y Tecnología de los Computadores**

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Grado de Luis Sánchez Velasco,

Informa:

que el presente trabajo, titulado:

***“Design of an embedded camera with an AX.25 transmitter ”***

ha sido realizado y redactado por el mencionado alumno bajo nuestra dirección, y con esta fecha autorizo a su presentación.

Granada, a 11 de Septiembre de 2017

A handwritten signature in black ink, appearing to read 'Andrés Roldán', with a long, sweeping horizontal stroke extending to the right.

Fdo. Andrés María Roldán Aranda




Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Grado se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 20 de junio de 2017

A handwritten signature in black ink, appearing to read 'Luis', with a long, sweeping horizontal stroke extending to the right.

Fdo. Luis Sánchez Velasco

A handwritten signature in black ink, appearing to read 'Andrés María Roldán Aranda', with a long, sweeping horizontal stroke extending to the right.

Fdo. Andrés María Roldán Aranda





# Design of an embedded camera with an AX.25 transmitter

Luis Sánchez Velasco

## PALABRAS CLAVE:

[Arduino](#), Empotrado, Firmware, Cámara, Diseño [PCB](#), [Altium](#), [AX.25](#).

## RESUMEN:

El proposito principal de este proyecto es desarrollar un producto completo capaz de leer varios sensores tanto meteorológicos como de posicionamiento, además de dos cámaras y enviar los datos resultantes usando el protocolo de radio [AX.25](#)

El proyecto intenta abordar este diseño como si de un producto de *hardware* para uso comercial se tratase, poniendo enfoque sobre las diferentes partes del diseño de *hardware* y estudiando capa parte por separado.

Se pretende además el uso de tecnologías emergentes en el mundo de la electrónica de *hobby* como son las [breakout boards](#) para facilitar el acceso a mucha gente de estas tecnologías.

# Design of an embedded camera with an AX.25 transmitter

Luis Sánchez Velasco

## KEYWORDS:

[Arduino](#), Embedded, Firmware, Camera, [PCB Design](#), [Altium](#), [AX.25](#).

## ABSTRACT:

The main purpose of this project is to develop a full product able to read data from several sensors to obtain positional and weather data, two cameras are also included. This data is then sent by the radio protocol [AX.25](#).

This project tries to tackle the design as if it was a commercial hardware product, placing the focus on the several stages of hardware design, studying every layer on its own.

The use of emergent technologies in the world of *hobby* electronics is pretended, making use of [breakout boards](#) to facilitate the access of many people to these technologies.

*All hope is gone.*



## *Agradecimientos:*

A mi familia y amigos por apoyarme durante estos años. Y a la gente aleatoria de internet.

## *Acknowledgments:*

To my family and friends for supporting me throughout this years. And to the random people of the internet.



# INDEX

Autorización Lectura	v
Autorización Depósito Biblioteca	vii
Resumen	ix
Dedicatoria	xi
Agradecimientos	xiii
Index	xv
List of Figures	xxi
List of Tables	xxv
Glossary	xxvii
Acronyms	xxix



<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	State of the Art . . . . .	2
1.1.1	Weather Balloons of today . . . . .	3
1.2	Motivation . . . . .	4
1.3	Objectives . . . . .	4
1.4	Methodology . . . . .	5
1.5	Project Structure . . . . .	7
<b>2</b>	<b>System Requirements</b>	<b>9</b>
2.1	Mechanical requirements . . . . .	9
2.2	Hardware requirements . . . . .	9
2.3	Software requirements . . . . .	10
<b>3</b>	<b>System Analysis</b>	<b>11</b>
3.1	Power Module . . . . .	12
3.1.1	Battery . . . . .	12
3.1.1.1	9 V batteries . . . . .	12
3.1.1.2	1.5 V AAA batteries . . . . .	14
3.1.1.3	Lithium batteries . . . . .	15
3.1.2	Comparative table . . . . .	16
3.1.3	Final conclusion . . . . .	16
3.2	Weather modules . . . . .	16
3.2.1	Temperature sensor . . . . .	16
3.2.2	Barometric sensor . . . . .	17
3.2.2.1	Market options, BMP180 . . . . .	18
3.3	Location modules . . . . .	18
3.3.1	Magnetometer . . . . .	19
3.3.1.1	Working principle of a MEMS Magnetometer . . . . .	19
3.3.2	Accelerometer . . . . .	21

3.3.2.1	Working principle of a MEMS Accelerometer . . . . .	21
3.3.3	LSM303 . . . . .	23
3.3.4	Gyroscope . . . . .	23
3.3.4.1	Working principle . . . . .	24
3.3.4.2	MPU6050 . . . . .	25
3.3.5	GPS system . . . . .	25
3.3.5.1	GPS overview . . . . .	25
3.3.5.2	GPS message format . . . . .	26
3.3.5.3	u-blox NEO-6 . . . . .	26
3.3.6	RTC . . . . .	26
3.4	Camera . . . . .	26
3.4.1	OV7670 overview . . . . .	27
3.5	Communication modules . . . . .	27
3.6	MCU . . . . .	28
<b>4</b>	<b>System Design</b> . . . . .	<b>29</b>
4.1	Electronic design . . . . .	29
4.1.1	PCB design . . . . .	31
4.1.1.1	Power budget . . . . .	43
4.1.1.2	LSF0108 voltage level translation . . . . .	43
4.2	Software design . . . . .	45
4.2.1	Activity diagram . . . . .	45
4.2.2	BMP180 . . . . .	46
4.2.3	LSM303 . . . . .	49
4.2.4	MPU-6050 . . . . .	50
4.2.5	TinyRTC / DS1307 . . . . .	51
4.2.6	GPS . . . . .	52
4.2.7	OV7670 - NO FIFO . . . . .	53

4.2.7.1	Setup . . . . .	54
4.2.7.2	Data collection . . . . .	55
4.2.8	OV7670-FIFO . . . . .	56
4.2.8.1	Setup . . . . .	57
4.2.8.2	Data collection . . . . .	57
4.2.9	HX-1 Data transmission . . . . .	59
4.2.9.1	AX.25 frame generation . . . . .	59
4.2.9.2	APRS frame . . . . .	59
<b>5</b>	<b>Verification, Testing and Manufacturing</b>	<b>61</b>
5.1	Camera testing . . . . .	61
5.2	Electronic testing . . . . .	63
5.3	Software testing . . . . .	65
5.3.1	BMP180 . . . . .	65
5.3.2	LSM303 . . . . .	66
5.3.3	MPU6050 . . . . .	66
5.3.4	DS1307 . . . . .	67
5.3.5	HX-1 Radio Module . . . . .	67
<b>6</b>	<b>Conclusions and future lines</b>	<b>69</b>
6.1	Future lines . . . . .	70
<b>A</b>	<b>Weather balloon physics</b>	<b>73</b>
A.1	Maximum payload weight . . . . .	74
A.2	Weather balloon velocity . . . . .	74
A.3	Weather balloon volume . . . . .	74
A.4	Applying it to the 400-8210 weather balloon with helium . . . . .	75
<b>B</b>	<b>Basic AX.25 documentation</b>	<b>77</b>
B.1	Frame structure . . . . .	77

B.1.1	Flag field . . . . .	78
B.1.1.1	Address field . . . . .	78
B.1.1.2	Control field . . . . .	78
B.1.1.3	PID field . . . . .	78
B.1.1.4	Information field . . . . .	78
B.1.1.5	Frame Check Sequence . . . . .	78
<b>C</b>	<b>Generating an AFSK signal with Arduino</b>	<b>79</b>
C.1	Timer Counter Setup . . . . .	79
C.2	PWM generation . . . . .	80
<b>D</b>	<b>DIY PCB manufacturing</b>	<b>83</b>
D.1	Transferring the stencil to the copper . . . . .	83
D.2	Removing the surplus copper . . . . .	85
<b>E</b>	<b>Project Budget</b>	<b>87</b>
E.1	Electronic Cost . . . . .	87
E.2	Software . . . . .	87
E.3	Human Resources . . . . .	88
	<b>References</b>	<b>91</b>



# LIST OF FIGURES

1.1	Trackduino banner	1
1.2	Basic scheme of the system.	2
1.3	GranaSAT's logo.	3
1.4	HX1 Radio Module	3
1.5	Design Method	6
1.6	Product development Gantt's diagram	8
3.1	Block model of the device	11
3.2	Block model of power system	12
3.3	Voltage versus time [1]	13
3.4	Mechanical properties of a 9 V battery [1]	13
3.5	Effective charge of the energyzer E92 [2]	14
3.6	Mechanical dimensions of the energyzer E92 [2]	14
3.7	Adafruit's USB LiIon/LiPoly charger [3]	15
3.8	Brokaw Cell [4]	17
3.9	Frequency response of a piezoelectric material [5]	18

3.10	Vectors involved in obtaining the board's heading . . . . .	19
3.11	World Magnetic Model [6] . . . . .	20
3.12	Internal structure of an Accelerometer . . . . .	22
3.13	Internal structure of a MEMS Gyroscope . . . . .	24
4.1	Final block model of the design . . . . .	30
4.2	Explanation of PCB rules . . . . .	31
4.3	PCB's 3D model . . . . .	42
4.4	Video of the PCB . . . . .	43
4.5	LSF basic setup . . . . .	45
4.6	LSF0108 usage in the schematics . . . . .	46
4.7	Main activity diagram . . . . .	47
4.8	Frame creation diagram . . . . .	47
4.9	Frame flushing diagram . . . . .	48
4.10	Timing diagram of the OV7670's output . . . . .	53
4.11	16 bit Timer/Counter of an ATMEga 2560 [7] . . . . .	54
4.12	Picture extraction of the OV7670 . . . . .	55
4.13	OV7670 + FIFO schematics [8] . . . . .	57
5.1	Interconnection between OV7670 and UNO . . . . .	62
5.2	Test board for camera OV7670 . . . . .	62
5.3	VGA quality image taken with the OV7670 . . . . .	63
5.4	Prototype of the PCB . . . . .	63
5.5	Full minted PCB board . . . . .	64
5.6	BMP180 test results . . . . .	65
5.7	Decoding an AFSK signal . . . . .	67
A.1	Balance of forces on the balloon. . . . .	73
D.1	Paper used to print the PCB . . . . .	83

---

D.2	Transferring the stencil to the PCB . . . . .	84
D.3	Submerging the PCB into water . . . . .	84
D.4	Resultant PCB from the stencil transfer process . . . . .	85
D.5	Submerging the board in the copper solvent solution . . . . .	85
D.6	Final prototype PCB . . . . .	86





# LIST OF TABLES

3.1	Battery comparison table . . . . .	16
3.2	Thermometer comparison table . . . . .	18
3.3	Barometer comparison table . . . . .	18
3.4	LSM303 main features [9] . . . . .	23
3.5	MPU6050 main features [10] . . . . .	25
3.6	NEO-6M GPS main features [11] . . . . .	26
3.7	OV7670 GPS main features [12] . . . . .	27
3.8	Radio modules comparative table [13] [14] [15] . . . . .	28
4.1	Design rules for chemical reaction PCB manufacture . . . . .	31
4.2	Device's power budget . . . . .	44
4.3	Register mapping for the BMP180 . . . . .	48
4.4	Register mapping for the LSM303 . . . . .	49
4.5	Register mapping for the MPU-6050 . . . . .	51
4.6	Register mapping for the DS1307 . . . . .	52

5.1	Data from LSM303 rotating through the Y axis . . . . .	66
5.2	Data from MPU-6050 rotating through the Z axis . . . . .	66
A.1	Relevant data for 400-8210 weather balloon . . . . .	75
B.1	Information frame construction . . . . .	78
E.1	Electronic cost of the PCB . . . . .	87
E.2	Software cost of product . . . . .	88
E.3	Human cost of product . . . . .	88
E.4	Component cost . . . . .	89

# GLOSSARY

**Accelerometer** A device that is able to measure acceleration, in its own coordinate frame..

**Altium** Paquete *software* EDA capaz de asistir el diseño de esquemáticos electrónicos y PCB, así como la simulación de sistemas electrónicos.

**Arduino** Open-source electronic prototyping platform enabling users to create interactive electronic objects featuring single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world..

**AX.25** A data link layer protocol derived from the X.25 protocol suite and designed for use by amateur radio operators. It is used extensively on amateur packet radio networks. AX.25 v2.0 and later occupies the data link layer, the second layer of the OSI model. It is mainly responsible for establishing connections and transferring data encapsulated in frames between nodes and detecting errors introduced by the communications channel. As AX.25 is a pre-OSI-model protocol, the original specification was not written to cleanly separate into OSI layers. This was rectified with version 2.0 (1984), which assumes compliance with OSI level 2..

**Barometer** A device that is able to measure atmospheric pressure..

**breakout board** A small PCB board that takes in (usually) an electrical component and make it easy to use by providing output pins and solving the more complex electronics in the breakout board itself..

**CubeSAT** A CubeSat (U-class spacecraft) is a type of miniaturized satellite for space research that is made up of multiples of  $10\times 10\times 10$  cm cubic units. CubeSats have a

mass of no more than 1.33 kilograms per unit, and often use commercial off-the-shelf (COTS) components for their electronics and structure.

**DSP** Microprocessor which has architecture for digital signal processing. The goal of DSP is measure and execute digital signal processing algorithms successfully.

**GranaSAT** GranaSAT is an academic project from the University of Granada consisting of the design and development of a picosatellite (Cubesat). Coordinated by the Professor Andrés María Roldán Aranda, GranaSAT is a multidisciplinary project with students from different degrees, where they can acquire and enlarge the necessary knowledge to front a real aerospace project. <http://granat.ugr.es/>.

**Gyroscope** A device that is able to measure or maintain a certain orientation. It is based on the principle of angular momentum..

**HX1** Radio transmitter module that offers 300mW RF output VHF. It has a very lower consumption and is well suited for narrow band applications. .

**Magnetometer** A device that is able to magnetism in direction or/and strength..

**QQVGA** Monitor image resolution of  $160 \times 120$  pixels.

**SD card** Secure Digital (SD) is a non-volatile memory card format developed by the SD Card Association (SDA) for use in portable devices.

**VGA** Monitor image resolution of  $640 \times 480$  pixels.

# ACRONYMS

**ADC** Analog to Digital Converter.

**AFSK** Audio Frequency-Shift Keying.

**APRS** Automatic Packet Reporting System.

**ASK** Amplitude Shift Keying.

**BCD** Binary Coded Decimal.

**BJT** Bipolar Junction Transistor.

**CDMA** Code Division Multiple Access.

**CMOS** Complementary MOS. Tecnología MOS Complementaria.

**DIY** Do it Yourself.

**EDA** Electronic Design Automation.

**GPS** Global Positioning System.

**I2C** Inter-Integrated Circuit.

**MCU** Micro-Controller Unit.

**MEMS** Microelectromechanical Systems.

**PCB** Printed Circuit Board.

**PWM** Pulse Width Modulation.

**RTC** Real Time Clock.

**SDR** Software Defined Radio.

**SPI** Serial Peripheral Interface.

**UART** Universal Asynchronous Receiver-Transmitter.

**UGR** Universidad de Granada.

**USB** Universal Serial Bus.

## CHAPTER

# 1

# INTRODUCTION

The following Bachelor Thesis end the studies of the degree in Telecommunication Engineering at the University of Granada. The aim of this project is to create an [Arduino](#) based tracker, able to send environmental data and pictures through a radio link.

This project also follows another project presented as a Bachelor Thesis by Fco. J. Lázaro Llorente [16] by improving the [PCB](#) realization and size and including the firmware. Both projects are an attempt to revive the Trackduino open source project [17], whose banner is show in figure 1.1, discontinued since 2014, by renewing the firmware and creating an improved [PCB](#) with additional modules.

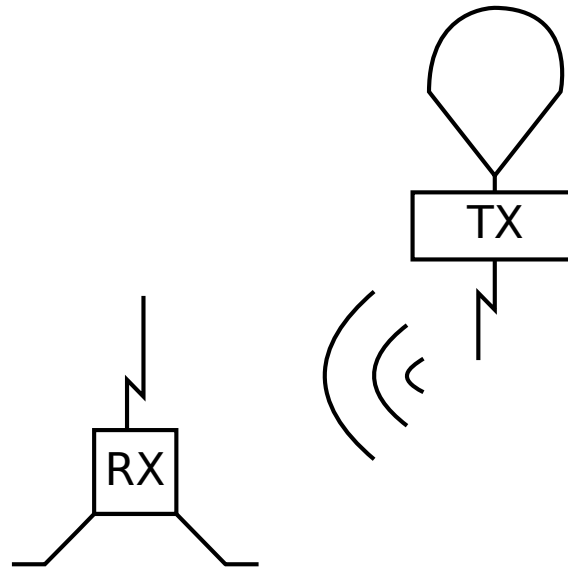


**Figure 1.1** – *Trackduino banner*

Let's begin this work by briefly explaining what a [GPS](#) tracker is and the basics of how it should work. A [GPS](#) tracker is essentially a device capable of being embedded into a vehicle of any type while sending data periodically to a master radio receiver. This data could be, for example, temperature, pressure or even images, an also, the device's current location. The communication is usually one sided, since the master will barely be in need to send messages to the the tracker.



A very basic representation of the situation can be seen in figure 1.2.



**Figure 1.2** – *Basic scheme of the system.*

It is possible to find many examples of such devices as for example the CanSAT I or the CrawSAT I[18] launched by the university of Louisiana, after all these devices are also used as a test platform for new devices included in standard satellites or CubeSATS. This last statement justifies the existence of this work since it is involved in the GranaSAT aerospace group.

GranaSAT is an aerospace group located at the University of Granada that aims to provide the student knowledge about the especial field and make them gain experience in this environment, whose ultimate goal is to build a fully functional CubeSAT and launch it. The group is mainly composed by students interested in the subjects and Master and Bachelor thesis. It is directed by prof. Andres Maria Roland Amanda. Logo is shown in figure 1.3.

During these years GranaSAT has participated in many programs such as the NEXUS/REX US program in which a stratospheric balloon was launched in order for the student to test their system in Kiruna (Sweden).

## 1.1 State of the Art

We can consider this project to be some kind of weather balloon, since the electronics and challenges faced in both products are very comparable to one another, even though the objectives could differ from the current project to a commercial weather balloon.



Figure 1.3 – *GranaSAT's* logo.

### 1.1.1 Weather Balloons of today

Weather balloons have become more and more a makers / [DIY](#) community thing due to the reduction of the price of the components needed to create them. Many of them can be found in several Makers web pages with an open source license, for example:

- [Tracksoar](#)
- [Trackuino](#)
- [Makers](#) magazine instructions and report on building one.
- [Instructables](#) guide to build one.

It can be seen from the previous sources that many of this projects have in common the use of the [APRS](#) signaling system on top of X.25 as data-link layer, instead of other modern protocols as IEEE 802.11. This is mostly because of [AX.25](#) being a more user friendly protocol, making small microcontrollers such as the avr2560 microprocessor by Atmel able to generate [AX.25](#) frames with ease. The protocol also offers error correction and frame counting [19].

The physical later usually consist of a radio layer at 144.8MHz using AFSK modulation. These choices were made due to the the fact that a base-band AFSK can be generated digitally with the microcontroller. This signal is then shifted up by a commercial radio module such as the Radiometrix [HX1](#) shown in figure 1.4.



Figure 1.4 – *HX1 Radio Module*

These modules often include a [GPS](#) system in order to track the balloon's current position, being the Ublox-GPS6MV2 one of the most used systems to the simplicity it offers, being able to interface with the microprocessor through a serial interface, requiring no setup.

## 1.2 Motivation

1

The aim of this work is to create a small and light enough device that carrying it away in a weather balloon or a small propulsion system could be achieved with ease. The system shall provide output pins so that external **breakout boards** could be connected and tested within the system. This work could facilitate the testing for many new modules yet to come deeply.

It also brings the student the opportunity of creating a useful **PCB** using modern techniques and learn about the manufacturing process of a current electronic system, subjects that are not easy to learn and usually requires the student to learn them as they create them.

The project is also part of the **DIY** community, since due to the use of mostly cheap components and not so hard electronics this project could be very easily modified /built/ recreate by a third person with little knowledge. All the schematics, gerbers, **PCBs** and firmware will put online for the community as open source hardware /software. Open Source Hardware Association's logo will be shown in figure 1.5a and Open Source initiative's logo will be shown in figure 1.5b.



(a) *Open Source Hardware Foundation's logo*



(b) *Open Source Initiative's logo*

## 1.3 Objectives

Even though some of the objectives were briefly in the previous section 1.2, here the objectives will be described in a more clear, concise and organized way. For this project to be considered successful the following requisites shall be fulfill:

1. Design a **PCB** where the modules can be connected and removed in a easy way to provide a easy to use test board.

2. Program a firmware that makes the board able to interact with the [Arduino](#) and send data to the base station.
3. Provide a firmware that enables the [Arduino](#) to interact and obtain data from the different sensors.

And therefore, in order to fulfill the objectives the student managing the project shall be able to:

1. Read technical documentation and be able to apply the specifics of each module to a electronic design.
2. Be able to read technical documentation to create a firmware able to interact with the module in a satisfactory way, given that 1 is fulfilled.
3. Have the ability to apply critical reasoning and the concepts learned during the degree.

## 1.4 Methodology

In order to design a system successfully a method must be follow, to ensure that time is dealt in a efficient way and that solutions are achieved by critical thinking through qualitative and quantitative research. The use of such design method will lead with more probability to:

- Achieve the optimal solution.
- Ease of transformation of the design specifications, leading to a more competent product as time passes.
- Achieve solutions that otherwise won't be accessible,
- A better understanding of the project as a whole.

In the present design method for this project, we will divide the process in four blocks:

1. **System Requirements**, providing specifications of how the system shall behave without attending technical data.
2. **System Analysis**, breaking down the system into bigger blocks to gain a better understanding of the system. Also selecting the optimal solution.
3. **System Design**, the actual implementation of what was studied in the system analysis.
4. **Test and Verification**, the analysis of what was designed in the system design and if fulfills the system requirements.

The design methodology followed during this project is presented in figure 1.5.

1

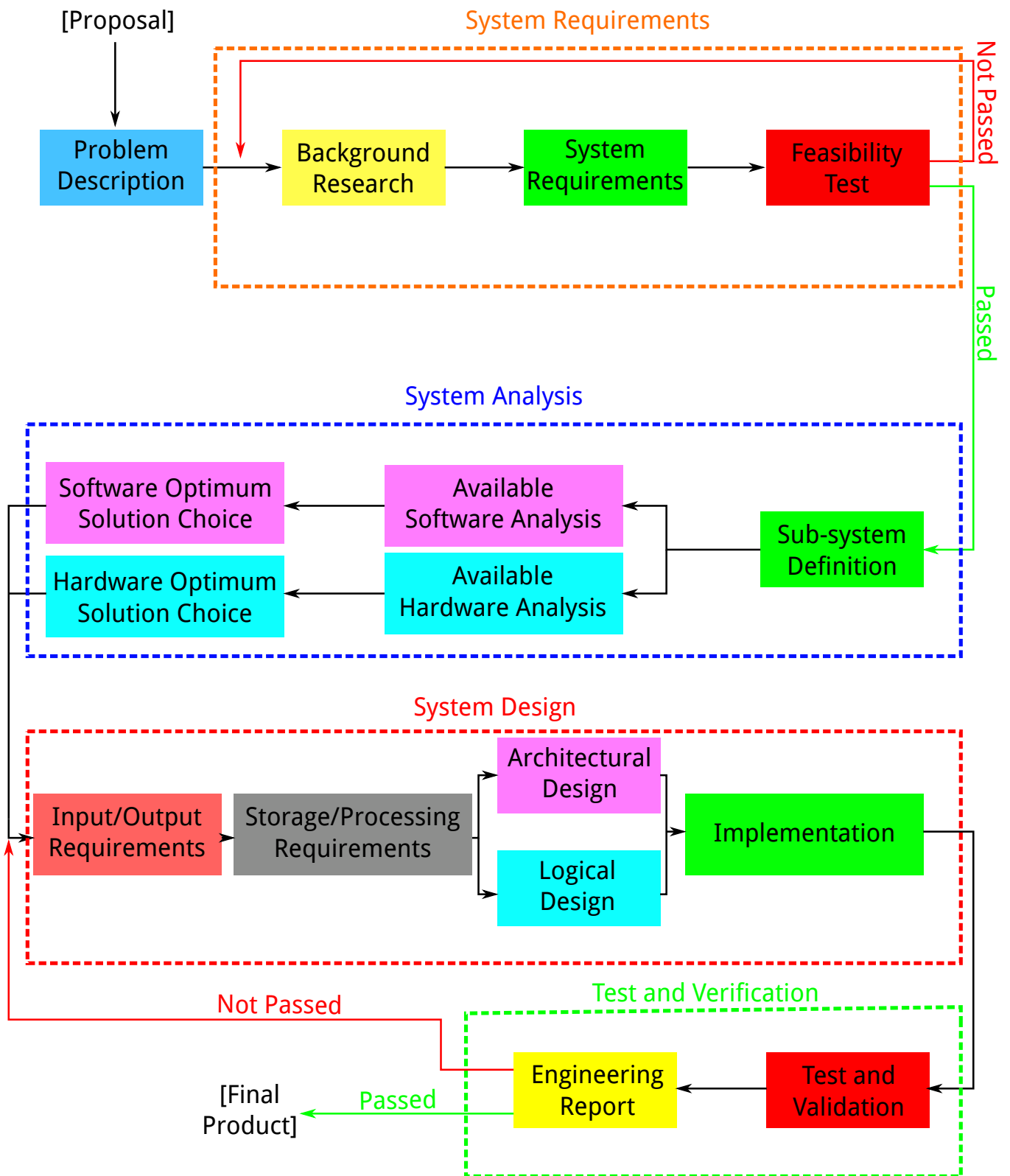


Figure 1.5 – Design Method

## 1.5 Project Structure

This project will be divided into 6 chapters and 5 appendix, this chapters will describe the procedure of designing and manufacturing the project, from the very basic idea of what is wanted and expected from the product to the in depths of the software and hardware created for it. The chapters forming this present document are:

1. **Introduction:** This chapter will introduce the project, from the conception of the idea to the planification of the work to complete the project. Figure 1.6 describe the Gantt diagram for the planification of the project.
2. **System Requirements:** This chapter will describe the essential features for the project to be considered valid.
3. **System Analysis:** During this chapter an overview of the system will be presented in big scale. This chapter will also approach the basic working principle of the sensors that will build the system.
4. **System Design:** In this chapter the designing of the system will be tackled, trying to explain the technical decisions made and the in depths of both the hardware and software in detail.
5. **Verification, Testing and Manufacturing:** This chapter will approach how the prototypes and the final product was manufactured. It will also approach the test made to verify the function of the different sensor and modules.
6. **Conclusion:** Sum up of the things learned and discovered during this work.
7. **Weather Balloon Physics:** Here a basic overview of how a regular weather balloon flies is presented.
8. **Basic AX.25 Documentation:** A little introduction of how a AX.25 frame is to familiarize the reader with the protocol.
9. **Generating an AFSK signal with Arduino:** This appendix will cover all the little tricks used to generate an AFSK signal with Arduino.
10. **Project Budget:** During this appendix the total cost of the project is analyzed.

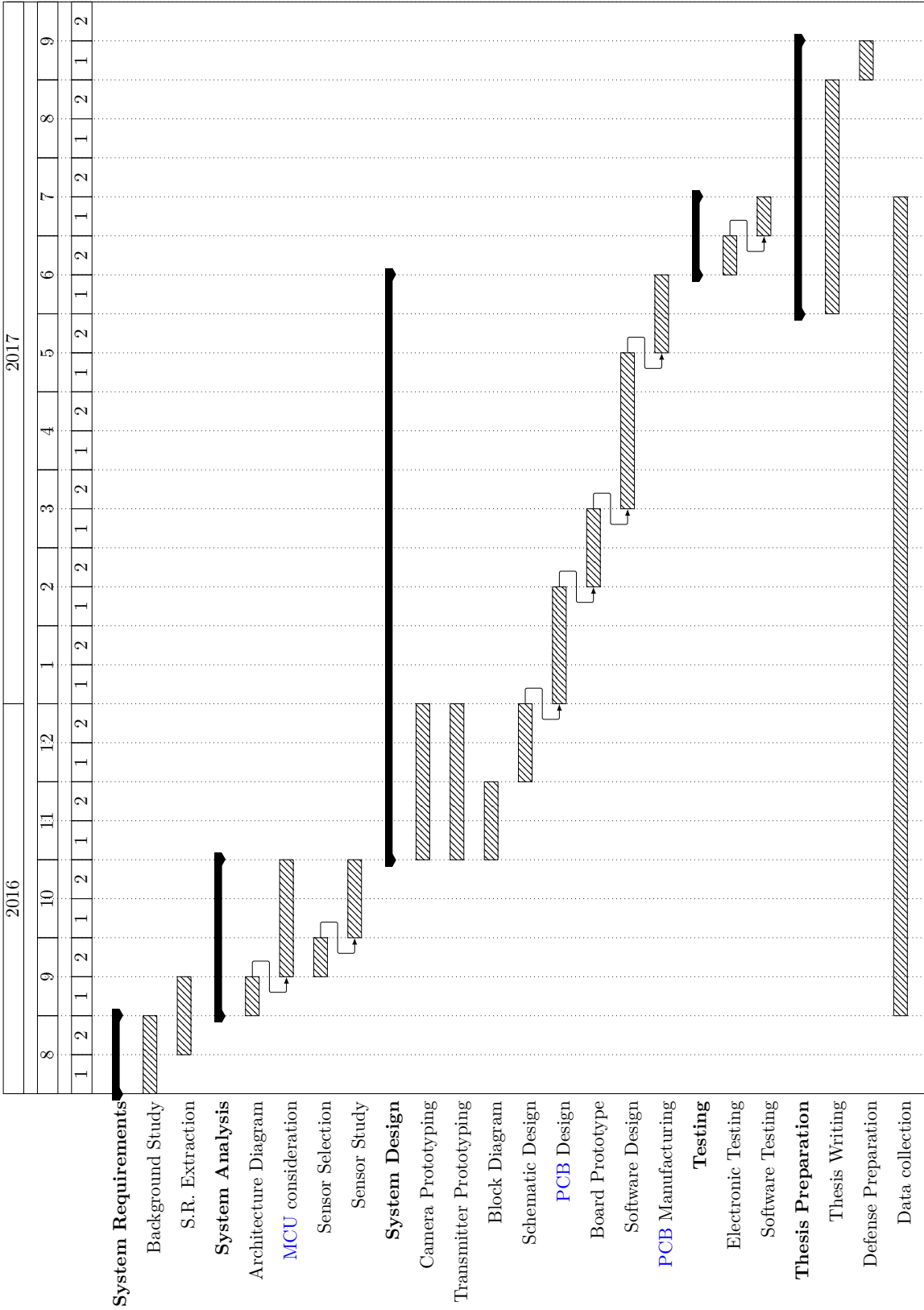


Figure 1.6 – Product development Gantt's diagram

## CHAPTER

# 2

# SYSTEM REQUIREMENTS

Having a great system requirements page are the key in a design and development of any complex system. They specify how a system should work and what features should be available on the system, as well as the task it should accomplish. These requirements are more about what the system shall do than how to do it. On this chapter three types of requirements will be differentiated: Mechanical requirements, Hardware requirements, Software requirements.

## 2.1 Mechanical requirements

1. The system shall have 4 mounting points for screws to go in.
2. The system shall be able to be carried by a weather ball on, that is, the total weight has to be less than 0.05kg. (see [A](#))
3. The board must have standard output pins for quick module exchange.

## 2.2 Hardware requirements

1. The design of a [PCB](#) capable of holding and interconnecting the board with the external modules.
2. A device capable of measuring temperature and pressure data.



3. The device shall be able to obtain data from a [Gyroscope](#) as well as an [Accelerometer](#) and a [Magnetometer](#).
4. The device's battery shall last for at least 2h30' (see [A](#)).
5. An external timing system with the current time shall be included.
6. Cameras shall be included in the device.
7. A [GPS](#) system shall be included to determine the global position of the board.
8. An external SD card must be included to hold all the data generated by the board.
9. All the data generated by the board must be transmitted via wireless to a base station.

### 2.3 Software requirements

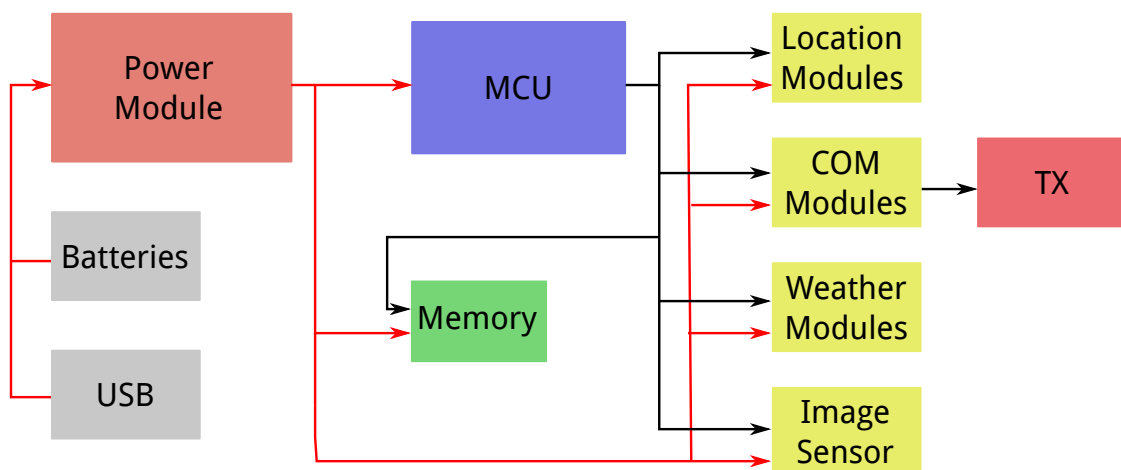
1. The software shall be integrated in the board.
2. The software must be able to operate all the modules mentioned in [2.2](#).
3. The software shall be able to retrieve pictures from the cameras and store them in the [SD card](#).
4. The software shall be able to produce an [AFSK](#) modulated signal in base band to feed into the radio modules.
5. The software shall be able to poll [GPS](#) data frequently.

## CHAPTER

# 3

## SYSTEM ANALYSIS

In this chapter the general overview to the system designed for chapter 2 will be provided. The different blocks composing our system will be described and justified for the utility they offer. Let's begin by taking a quick look at figure 3.1, in which a basic block model is proposed to solve the requirements in chapter 2.



**Figure 3.1** – *Block model of the device*

The proposed solution uses a MCU as the main *director* of the board, connected directly to all the modules, this will simplify the design quite a lot, since the code to control each

module can be written and tested in parallel and then put together very easily. The main drawback of this design is that the high number of modules required will require as well a better MCU than an hypothetical design with more computational power e.g. one where the cameras, weather modules and the communication modules are driven by an independent MCU and then sent to a *leader* MCU.

Each independent block will be discussed more in depth below.

### 3.1 Power Module

This block contains all the power related electronics, from the batteries to the voltage conversion units. This system shall be able to be charged or easily replaced when the power goes low, as well as work connected to an external power supply (mainly for testing purposes). This block shall be also able to provide 5 V and 3.3 V for the modules that require it. To solve this problem the block diagram in figure 3.2 is presented, it contains a simple solution with a battery, a battery protection circuit a DC-DC boost converter to 5 V and a regulated power supply from 5 V to 3.3 V.

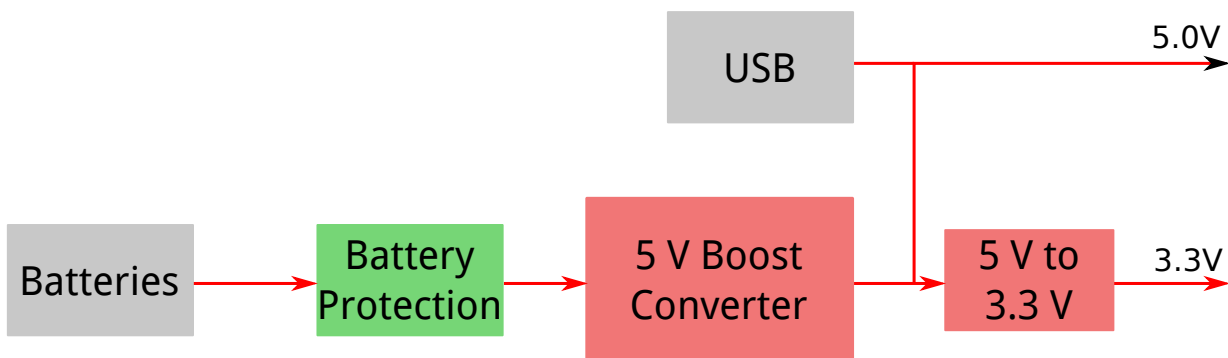


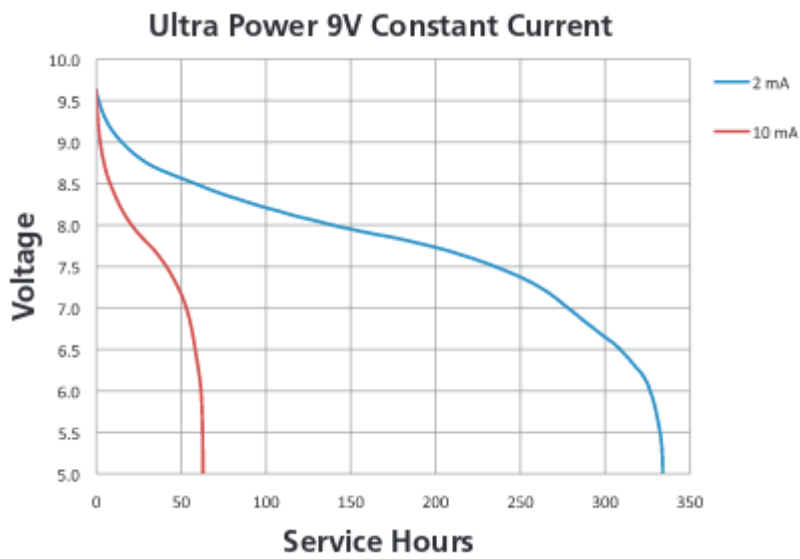
Figure 3.2 – Block model of power system

#### 3.1.1 Battery

Several options shall be analyzed in order to choose the most appropriate portable battery for the project. 9 V batteries, AAA alkaline and lithium-ion batteries will be discussed below.

##### 3.1.1.1 9 V batteries

These are very common batteries, really easy to find and buy for a reasonable price. Since all 9 V batteries are not the same, but hold very similar properties a particular one will be elected, the Duracell Ultra Power 9 V MX1604. From datasheet [1] we can obtain the total charge held by the battery by looking at figure 3.3:

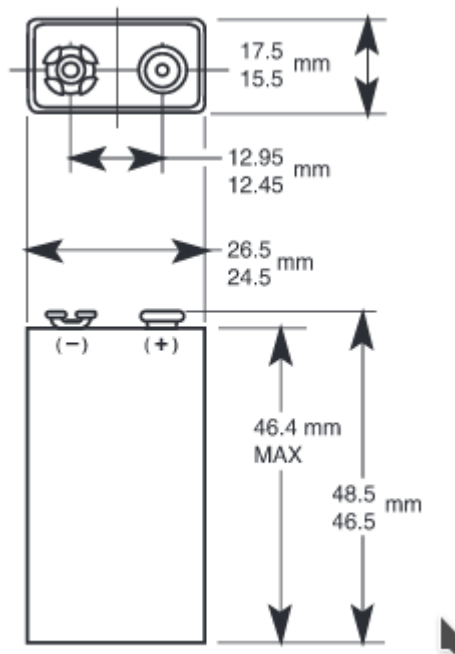


**Figure 3.3** – Voltage versus time [1]

From where we can see that:

$$\text{Charge} = 10\text{mA} \times 60\text{h} = 600\text{mAh} \quad (3.1.1)$$

Below, the mechanical model can be seen in figure 3.4



**Figure 3.4** – Mechanical properties of a 9 V battery [1]

### 3.1.1.2 1.5 V AAA batteries

As the other battery, it is a really simple to obtain cell. The mayor drawback of implementing this type of solution is that at least two cells will be needed to obtain 3V for the step up converter to work properly. As a model of this set of batteries we will take a look at the energyzer E92 [2]. This time the manufacturer will provide us directly with the total effective charge for different consumptions at figure 3.5.

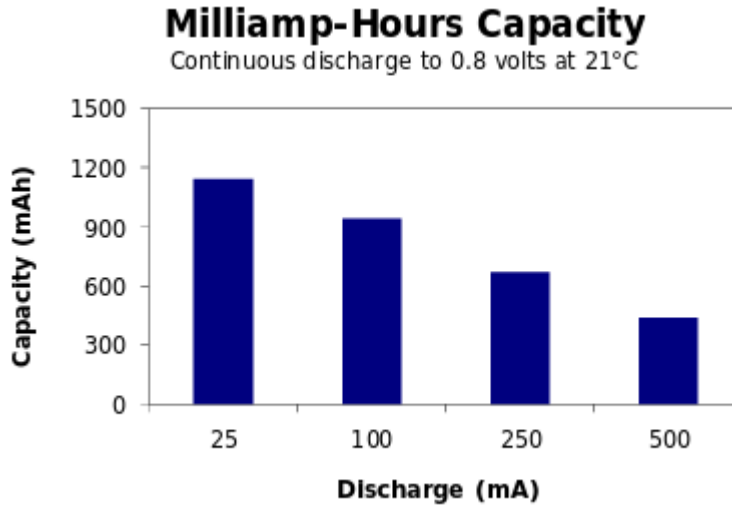


Figure 3.5 – Effective charge of the energyzer E92 [2]

A fair assumption is a consumption of 1 A (500 mA per cell) due to the amount of modules we have, thus leaving us with a effective charge of 1200 mAh. Figure 3.6 how us the mechanical dimensions.

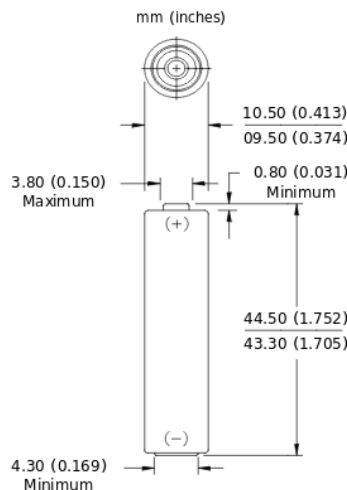


Figure 3.6 – Mechanical dimensions of the energyzer E92 [2]

### 3.1.1.3 Lithium batteries

Lithium batteries are the current <sup>1</sup> powerhouse for the modern digital revolution. These batteries come in many shapes and capacities due to flexible form factor [20], these batteries also present the highest energy density amongst all the other batteries mentioned before. They are also easily rechargeable and have no memory effect. Despite having all the benefits commented before, these batteries are really unstable and can cause serious harm, so a protection circuit is required [21].



As a battery example we can take the [easylander's EL-103050](#), the battery size is T=10 mm W=30 mm L=50 mm, more than reasonable for a small PCB, and the total charge equals 1350 mAh <sup>2</sup>, the protection circuit is already inside the battery itself and therefore we can interface it directly to the converter, but an external breakout board can be added such as the [Adafruit's USB LiIon/LiPoly charger](#) (figure 3.7) to add more protection and a charging USB port.

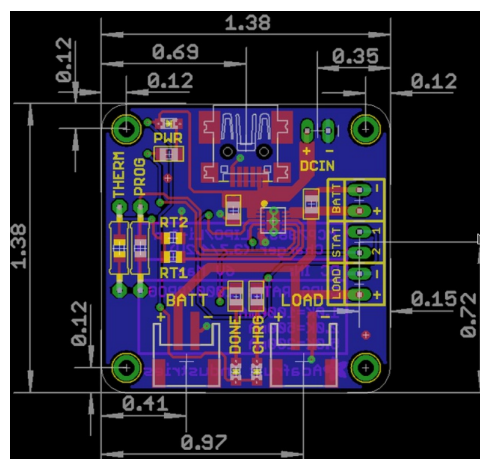


Figure 3.7 – Adafruit's USB LiIon/LiPoly charger [3]

<sup>1</sup>September 6, 2017

<sup>2</sup>These values were found directly on their site and therefore they are not really trustworthy

### 3.1.2 Comparative table

In order to choose the most appropriate battery a comparative table shall be made highlighting all the relevant features for our design, them being:

- Weight, the board will be raised by a balloon, the maximum weight for the whole construction is detailed in section [A.1](#)
- Size, the size of the board shall be kept as small as possible in order to reduce production cost.
- Energy, the board shall be able to operate for as long as possible.
- Price, since this is also a hobby/[DIY](#) project cheap material will be preferred.

	Weight (g)	Length (mm)	Wide (mm)	Height (mm)	Charge (mAh)	Price (€)
MX1604	45	48.5	26.5	17.5	600	5
Energizer E92	$2 \times 11.5 = 23$	44.5	$2 \times 10.5 = 21$	10.5	$2 \times 600 = 1200$	1.37 <sup>3</sup>
EL-103050	22	50	30	10	1350	4.17

**Table 3.1** – Battery comparison table

### 3.1.3 Final conclusion

By looking at table [3.1](#), where a comparison of the available power sources is presented, we can conclude that the most efficient battery and easy to use (AAA cells need a plastic holder to keep them in position) is the lithium battery plus the possibility of adding an [USB](#) charger for the battery opens a lot of space for extra design without adding much space or weight to the final [PCB](#) since the extra [breakout board](#) can be mounted on the battery itself.

## 3.2 Weather modules

The weather modules shall contain at least a [Barometer](#) and a thermometer on board in order to obtain data on the flight, the required modules will be now briefly described:

### 3.2.1 Temperature sensor

IC's temperature sensor usually relay on a band gap voltage reference [\[4\]](#), a circuit that maintains a stable voltage regardless of the ambient temperature, usually the “Brokaw Cell” which relies on the temperature dependence of the PN junction in a [BJT](#) transistor. The circuit representing the “Brokaw Cell” can be seen in figure [3.8](#), the temperature is obtained

by measuring  $V_{be}$  in both transistor, then equation 3.2.1 is applied.

$$\Delta V_{be} = \frac{KT}{q} \cdot \left( \frac{I_{C1}}{I_{C2}} \right) \quad (3.2.1)$$

Where  $\frac{I_{C1}}{I_{C2}}$  in a circuit that forces  $I_{C1}$  and  $I_{C2}$  to have a ratio 1 :  $N$  gives:

$$\Delta V_{be} = \frac{KT}{q} \cdot (N) \quad (3.2.2)$$

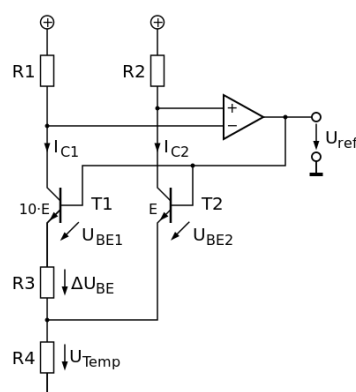


Figure 3.8 – Brokaw Cell [4]

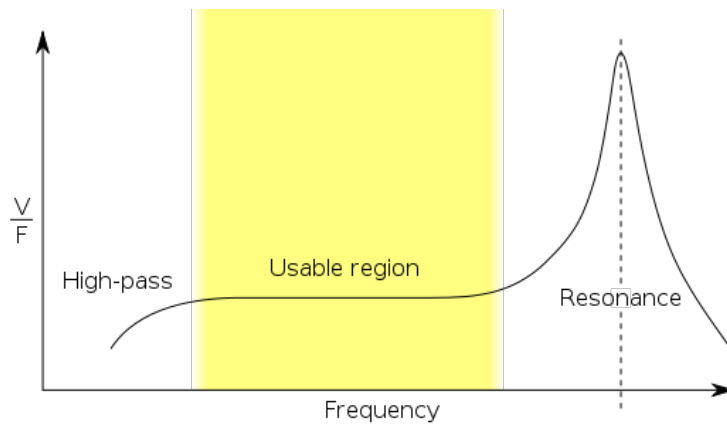
Then the analog measurement unit is directly interfaced to an ADC for analysis and obtainability.

### 3.2.2 Barometric sensor

A **Barometer** is a device that is able to measure atmospheric pressure. This data can for example be used to forecast short term changes in the weather or to determine the current altitude by comparing it to the exponential atmosphere model. The utilities presented by this device can place it in both location and weather modules.

In order to manufacture an integrated **Barometer** piezoelectric materials are very well used, these materials electrical charge changes proportional to the pressure applied to it creating the electro-mechanical dependence, this variation in charge is measured by, usually, an **ADC** to obtain and process the data. The frequency response of piezoelectric materials shall also be considered, since we are measuring the change in pressure, not the pressure itself, needing a calibration every time the device resets, and also as can be seen in figure 3.9, very slow changes in the pressure will not be properly detected by the IC.





**Figure 3.9** – Frequency response of a piezoelectric material [5]

### 3

#### 3.2.2.1 Market options, BMP180

Once the working principle of the silicon thermometer and the [Barometer](#) are understood it is time to search the available [breakout boards](#) available and compare them in order to choose the best option. I personally chose buy a board that contains both sensor since they are widely available and really precise temperature/pressure data are not needed. Results are presented in table 3.2.

	Temperature range (°C)	Sensibility ( $\pm V$ )	Voltage supply (V)	Consumption ( $\mu A$ )	Communication	Price (€)
MPL3115A2 [24]	-40 - 85	1	1.95-3.6	40	I2C	8.45
BMP280 [25]	-40 - 65	1	1.75-3.6	2.7 @ 1Hz	I2C/SPI	8.45
BMP180 [26]	-40 - 65	1	2-3.6	5 @ 1Hz	I2C	Disc

**Table 3.2** – Thermometer comparison table

	Pressure range (hPa)	Sensibility ( $\pm$ hPa)
MPL3115A2 [24]	500 - 1100	4
BMP280 [25]	200 - 1100	1
BMP180 [26]	200 - 1100	1

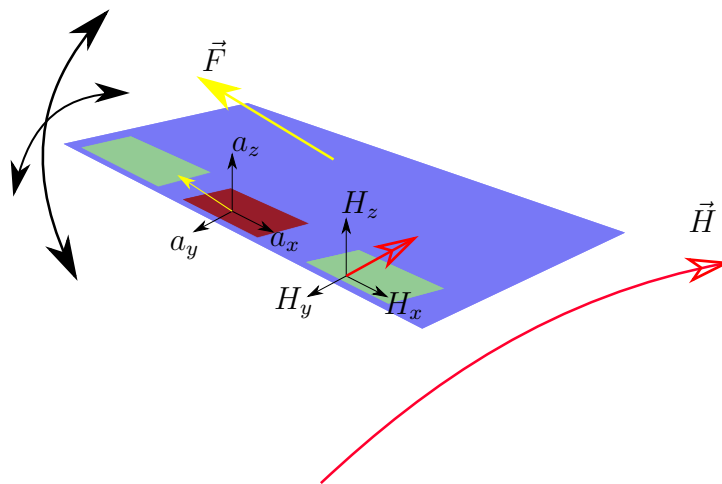
**Table 3.3** – Barometer comparison table

From tables 3.2 and 3.3 we can see that the best choice is the Adafruit's BMP280 [breakout board](#), however BMP180's [breakout board](#) will be used since it is the one available at the lab.

### 3.3 Location modules

Location modules refer to all modules whose main purpose in the [PCB](#) is to obtain data referred to the current position, speed and heading of the system during flight time. In order to obtain this data it will be needed:

- A compass, to obtain the heading towards an absolute reference, implemented as a [Magnetometer](#).
- An [Accelerometer](#), to obtain the current acceleration.
- A [Gyroscope](#), obtaining the angular velocity and the rotation of the board, critical for the other systems to work properly since their reference plane is on the board.
- [GPS](#) receiver, to obtain the global position of the board.



**Figure 3.10** – Vectors involved in obtaining the board's heading

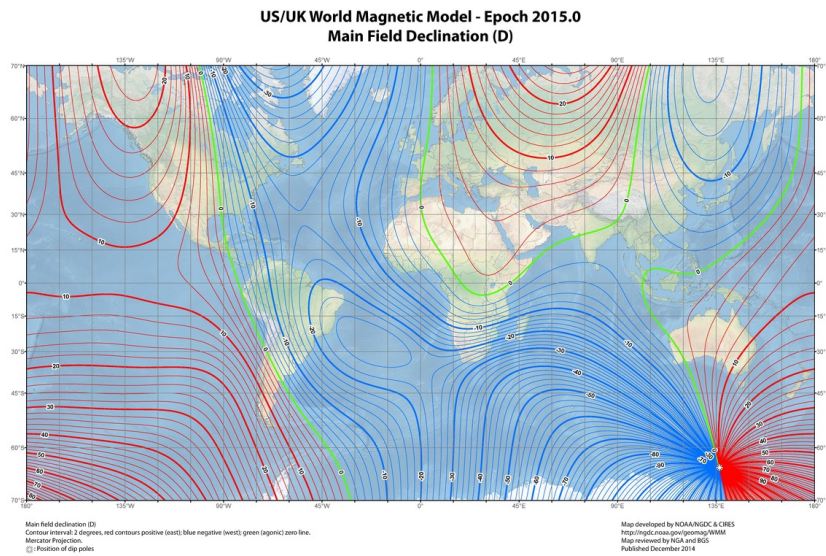
### 3.3.1 Magnetometer

The [Magnetometer](#) is a device capable of measuring the magnetism of material, in both direction and strength, in our design this sensor will be used as a device to calculate the absolute angle of rotation in the azimuth angle of the [PCB](#) by comparing it to the Earth's magnetic field. This magnetic field can then be compared using the **World Magnetic Model** to properly point to the north. The deviation in the azimuth angle between the Geographical North and the Magnetic North is called **declination**, meanwhile the deviation in the zenith angle is called **inclination**. In figures 3.11a and 3.11b both inclination and declination maps are displayed.

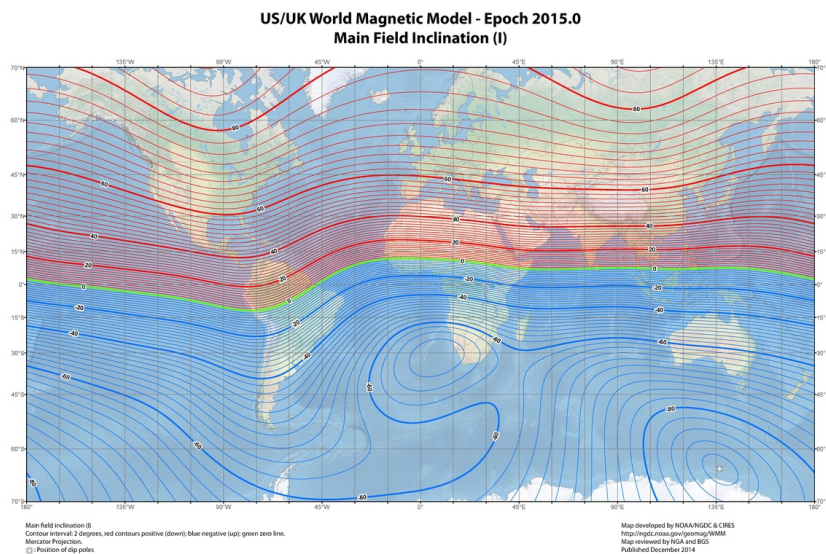
As it can be seen in figure 3.11a, currently in Spain the declination is very small making it such that no real correction is needed, simplifying the later firmware.

#### 3.3.1.1 Working principle of a [MEMS Magnetometer](#)

Most of the current [MEMS Magnetometer](#)/compass are based on the detection of the rotation in a small metal bar due Lorentz force, this rotation is then measured by the



(a) World Magnetic Model declination map of 2015 to 2020 [6]

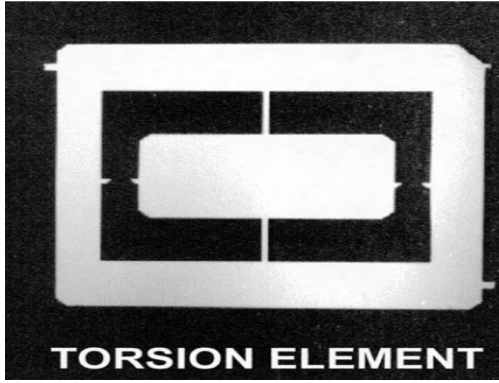


(b) World Magnetic Model inclination map of 2015 to 2020 [6]

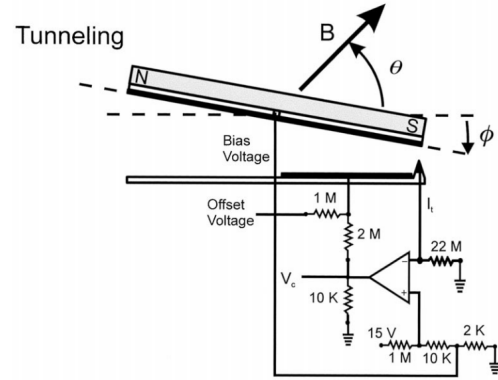
**Figure 3.11** – World Magnetic Model [6]

necessary feedback to maintain a constant tunneling current through the device, achieving a resolution of  $0.3 \text{ nT}/\sqrt{\text{Hz}}$  [27], below in figure 3.12a the schematic is displayed.

When the device is first powered the gap between the bar and the tunneling nail is large, so  $V_c$  is set. The bar, due to the introduced voltage a torque is created causing the bar to spin thus reducing the tunneling gap, until the desired tunneling current is achieved, point where the bar will not move, unless the local magnetic field changes, in which the circuit reacts, varying  $V_c$ . The following formula can be deduced relating the angle of the bar with



(a) Schematic diagram of the micro machined magnetic-field tunnel sensor [28]



(b) Photograph of a micro machined part of a magnetic-field tunnel sensor [28]

with the control voltage,  $V_c$  [28]:

$$V_t = \sqrt{\frac{2\tau_{opposing}S_0^2}{k\epsilon r_e A}} \quad (3.3.1)$$

where:

$$\tau_{opposing} = k_t\phi_0 - m(B_0 \sin \theta_0 + B_1 \sin \theta_1) \quad (3.3.2)$$

### 3.3.2 Accelerometer

An **Accelerometer** is a device that is able to measure proper acceleration, that is, the acceleration relative to a free fall, e.g. an **Accelerometer** in Earth in rest will measure an upwards force of  $9.82m \cdot s^{-2}$ , since it is measured by a “viewer” that is free falling, if instead that **Accelerometer** is left free falling it will measure  $0m \cdot s^2$ . This sensor is extremely useful since it can be used to effectively know when the balloon lifting the **PCB** has reached its maximum and the device is going to fall immediately. It can be used to track wind currents since most likely the balloon will follow them.

#### 3.3.2.1 Working principle of a **MEMS Accelerometer**

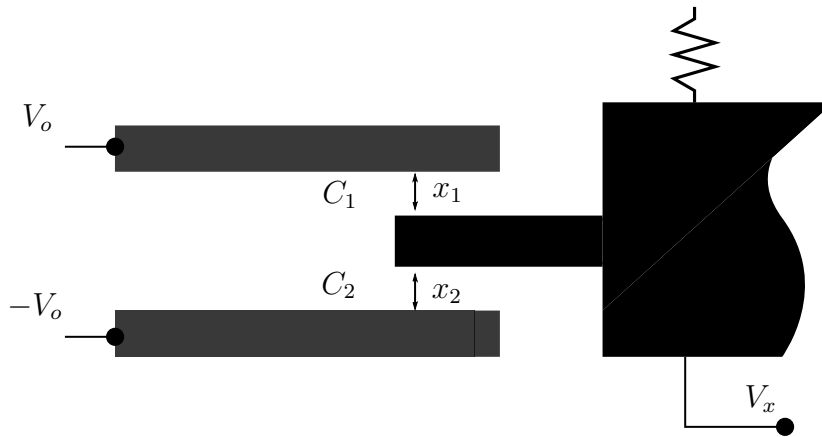
A **MEMS Accelerometer** is mainly formed by a spring holding movable plates, as it can be seen in figure 3.12, this spring will react to any force applied to it (hook’s law) thus modifying the capacitance formed by capacitors  $C_1$  and  $C_2$ . We can obtain the expression for the capacitance for each capacitor by:

$$C_1 = \epsilon_0\epsilon \frac{A}{x_1} = \epsilon_A \frac{1}{x+d} \quad (3.3.3)$$

$$C_2 = \epsilon_0\epsilon \frac{A}{x_2} = \epsilon_A \frac{1}{x-d} \quad (3.3.4)$$

Where:

- $\epsilon_A$  is the dielectric constant of the capacitor times the effective area.
- $d$  is the distance between the two external plates
- $x$  is the distance between the center of the plate and plate  $C_1$  or  $C_2$  due to accelerations.



**Figure 3.12** – Internal structure of an *Accelerometer*

Then by defining  $C_0$  as the capacitance between the two external plates, we can operate in the following way:

$$C_1 = C_0 - \Delta C \quad (3.3.5)$$

$$C_2 = C_0 + \Delta C \quad (3.3.6)$$

$$C_2 - C_1 = 2\Delta C = 2\epsilon_A \frac{x}{d^2 - x^2} \quad (3.3.7)$$

$$\Delta C x^2 + \epsilon_A x - \Delta C d^2 = 0 \quad (3.3.8)$$

Where a non-algebraic equation is obtained. This equation can be simplified by assuming that the distance  $x$  will be very small, allowing us to neglect the term  $\Delta C x^2$ , obtaining:

$$x \approx d \frac{\Delta C}{C_0} \quad (3.3.9)$$

Now if  $V_0$  is set with a high enough frequency it holds true that:

$$(V_x + V_0)C_1 = (V_x - V_0)C_2 \quad (3.3.10)$$

$$V_x = \frac{x}{d} V_0 \quad (3.3.11)$$

And then by applying Hooke's law:

$$V_x = \frac{ma}{kd} V_0 \quad (3.3.12)$$

Where:

- $m$  is the mass of the intermediate plate and what holds it.
- $k$  is the characteristic constant of the spring.
- $d$  is the distance between the external plates.
- $V_0$  is the applied voltage.

### 3.3.3 LSM303

The available module at the lab is the LSM303, it includes an [Accelerometer](#) and a [Magnetometer](#), both working in a similar way as explained in sections 3.3.2.1 and 3.3.1.1. These analogical values are then read by an ADC and obtained by a control logic that will feed them to the [I2C](#) port on request [9]. The main features of this device are described in table 3.4.

LSM303	
Voltage supply	2.5-3.3V
Current consumption (normal mode)	0.83mA
Current consumption (power down mode)	3 $\mu$ A
Magnetic field scale	$\pm 1.3$ to $\pm 1.8$ Gauss
Acceleration scale	$\pm 2$ / $\pm 4$ / $\pm 8$ g (selectable)
Resolution	16 bit (both magnetic field and acceleration)
Communication	<a href="#">I2C</a>
Price	<a href="#">14.95 \$</a>

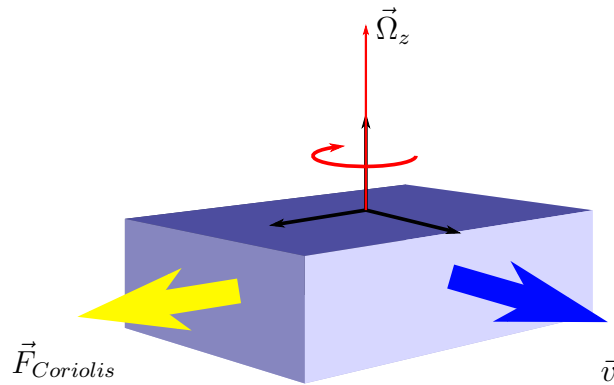
**Table 3.4** – *LSM303 main features [9]*

### 3.3.4 Gyroscope

A [Gyroscope](#) is an element that measures the orientation by keeping its own position based on the conservation of the angular momentum. The main purpose of including this device in the [PCB](#) is to keep track of the orientation of the device with more accuracy than the [Magnetometer](#) could have. This information could be used for example to select the appropriate time to take a picture.

### 3.3.4.1 Working principle

For this particular device the main used method to obtain the rotating velocity of the sensor itself is through the use of a vibrating piezoelectric material. This material will move back and forth with a known velocity that is related to the vibrating frequency of the material itself. This material is put in suspension by a system of springs so when a rotation event happens, Coriolis force will be applied to the material thus pushing the springs. The force created by this device can be measured with a similar system used in the [Accelerometer](#). The obtained force is directly proportional to the angular velocity the suspended material is exposed to, a graphic is displayed in figure 3.13.



**Figure 3.13** – Internal structure of a *MEMS Gyroscope*

The vibrating material will have a phase velocity of  $\omega_r$ , so its position and velocity at a given time can be obtained through the following equations 3.3.13 and 3.3.14:

$$p(t) = X_{ip} \sin \omega_r t \quad (3.3.13)$$

$$v(t) = \frac{dp(t)}{dt} = X_{ip} \omega_r \cos \omega_r t \quad (3.3.14)$$

Then, when rotating, the vibrating material experiences a force in the direction of the Coriolis force, described as:

$$a_c = 2(\Omega \times v) \quad (3.3.15)$$

And since both this material are perpendicular we can obtain the Coriolis force suffered by the material by combining eqs 3.3.15, 3.3.14 and multiplying by the system's mass, obtaining:

$$F_c = 2m\Omega X_{ip} \omega_r \cos \omega_r t \quad (3.3.16)$$

Where:

- $F_c$  is the resultant force

- $m$  is the system's mass
- $\Omega$  is the angular velocity
- $X_{ip}$  is the vibration amplitude of the system
- $\omega_r$  is the vibrating frequency

### 3.3.4.2 MPU6050

In the designed board a [breakout board](#) containing the [Gyroscope MPU6050](#) will be used, since is the one available at the laboratory, to sum up the most relevant data for implementation table 3.5 is presented. This board also contains an [Acceleromter](#).

MPU6050	
Voltage supply	2.375-3.46V
Current consumption (normal mode)	3.9mA
Current consumption (power down mode)	10 $\mu$ A
<a href="#">Gyroscope</a> scale	250 to 2000 °/s (adjustable)
Resolution	16 bit
Communication	<a href="#">I2C</a>
Price	<a href="#">2.81 €</a>

**Table 3.5** – MPU6050 main features [10]

### 3.3.5 GPS system

[GPS](#) is probably one of the most useful modules included in the [PCB](#), it gives the ability to geolocate the [PCB](#) during flight time. This let us know where exactly is the data taken from and will let us collect again the board once the flight time is completed.

#### 3.3.5.1 GPS overview

In order to geolocate a device a large amount of satellites are set in space, this satellites will broadcast their current location and the time when the location information was sent,  $[x_i, y_i, z_i, s_i]$ . By having the time when the frame was sent, it is possible to obtain the transit time as  $t_i - b - s_i$ , where  $t_i$  is the local time of the device when the frame was received and  $b$  is the bias from the more precise clock of the [GPS](#) satellite to the local clock. If we assume  $b$  not to vary that much between receiving [GPS](#) frames in a short amount of time, the equation system 3.3.17 can be solved with four satellites or more.

$$(x - x_i)^2 - (y - y_i)^2 + (z - z_i)^2 = ([t_i - b - s_i]c)^2, i = 1, 2, \dots, n \quad (3.3.17)$$



### 3.3.5.2 GPS message format

Each GPS system is continuously broadcasting a navigation message, usually at 1575.42 MHz (public) or 1227.60 MHz (military), encoded using CDMA so each satellite can be individually distinguished. Each frame consists of 1500 bits sent at 50 bits/s, these frames are then subdivided in 5 sub-frames. The first sub-frame contains the satellite clock information, the second and third subframes contains the precise satellite orbit, known as the ephemeris. Subframes 4-5 are sent in 25 different frames, that means that to receive the *full frame*, the receiver should listen to 25 different frames. In this subframes the almanac is contained, this provides information about all the other satellites, which is not currently necessary.

### 3.3.5.3 u-blox NEO-6

The available GPS at the lab is the u-blox neo-6, it is a basic module that is accessible through UART and will provide us with the basic GPS information, as the main characteristics, they will be shown in table 3.6

NEO-6M	
Voltage supply	2.7-3.6 V
Current consumption	47 mA
Antenna gain	50 db
Communication	UART and SPI
Price	10.19 €

**Table 3.6** – NEO-6M GPS main features [11]

### 3.3.6 RTC

An RTC is an external clock that keeps the actual time (in hours, minutes and seconds) and will provide our device with an external, more trustworthy, time reference, in order to be able to trigger some events on time as well as send a time stamp in the broadcasted frames by the communication module.

The working principle is really simple, it is basically a chip that is clocked from a 32.768 kHz crystal oscillator, this chip will count every  $2^{15}$  a second.

## 3.4 Camera

To complete the project an image sensor/camera will be required. One of the cheapest camera modules will be used, the OV7670 image sensor. This camera is based on CMOS technology to retrieve the image, this is basically a matrix of light sensible CMOS transistors that can be individually addressed. In order to obtain color information, individual coloring

filtering is added through a lens to each transistor, forming a *Bayern filter*, resulting in each pixel containing only information about one of the three fundamental colors, red, green and blue. Then with this information an interpolating algorithm can be applied to obtain the full image with three bits per pixel.

### 3.4.1 OV7670 overview

The camera/image sensor that will be used is the OV7670 module. The sensor itself contains the image array that is read by an analog processing unit that will do the analog to digital conversion as well as adjusting the exposure and the gain, the digital signal generated is then fed into a [DSP](#) to complete operations as de-noise, white balancing and black pixel correction. After that the pixel is sent to a FIFO memory, and then to the output 8 bit video port. The main features of this sensor are listed in table [3.7](#)

OV7670	
Voltage supply	2.45-3 V
Current consumption (active)	18 mA
Current consumption (standby)	18 mA
Image resolution	VGA, QVGA, QQVGA
Color encoding	RGB, YUV, YCbCr
Communication	<a href="#">I2C</a> for camera setup and an 8 bit parallel port for image acquisition
Price without FIFO	<a href="#">4 €</a>
Price with FIFO	<a href="#">7.95 €</a>

**Table 3.7** – *OV7670 GPS main features* [[12](#)]

## 3.5 Communication modules

The board shall be able to broadcast messages to the outside, to be received by the ground station and retrieve the data. For that reason a device capable of sending this data is required. This data will be encapsulated into an [AX.25](#) frame and modulated using [AFSK](#) for it to be set to the transmission frequency by the radio module itself. The main radio module will be composed by an HX1-144.8 by Radiometrix, another radio transmitter, using the FS1000A module, will be available in case of a failure of the first one. A comparatory table is shown in table [3.8](#).

Additionally another radio module is available the HC-12, the main difference with the previous ones is that this module communicates with the [MCU](#) by sending AT commands through a serial port.

	HX1-144.8	FS1000a	HC-12
Voltage supply	5 V	3.5-1.2 V	3.2 V-5.5 V
Current consumption	140 mA	3-10 mA [29]	200 mA
Operating frequency	144.8 MHz	433 MHz	433.3 MHz
Accepted modulations	Anything with a bandwidth less than 25 kHz	ASK [29]	-
Data bit rate	10 kbps	4 kbps	115 kbps
Range	10 km	40-100 m [29]	10 km
Transmit power	300 mW	10 mW	100 mW
Price	50.99 \$	2.5 €	7.99 €

**Table 3.8** – Radio modules comparative table [13] [14] [15]

### 3.6 MCU

The **MCU** will be the main board controller, coordinating all the board modules as expected, it can be said to be the *brain* of the **PCB**. Since this project needs to be widely available for all the people a well known board will be used, an **Arduino** ATmega2560. The justification for the **Arduino** Mega instead of the UNO (ATmega328P) is the number of I/O pins and hardware modules needed, listed below.

- SD card: SPI pins + 1 I/O pin.
- HX1: 2 I/O pins (1 **PWM**).
- HC-12: 1 **UART** + 1 I/O pins.
- FS1000A: 1 I/O pin.
- LSM303: 1 **I2C**.
- MPU6050: 1 I/O pin + 1 **I2C**.
- U-BLOX 6M GPS: 1 **UART**.
- **RTC**: 1 **I2C**.
- BMP180: 1 **I2C**.
- Camera OV7670 without FIFO: 12 I/O pins (1 **PWM**) + 1 **I2C**.
- Camera OV7670 with FIFO: 14 I/O pins + 1 **I2C**.

Making up a total of 1 **I2C** (since it's shared), 2 **UART** and 30 I/O pins which is beyond of what the ATmega32P can provide us with.

## CHAPTER

# 4

# SYSTEM DESIGN

During this chapter the system design and implementation details will be addressed. The following design is made to give a solution to the system requirements exposed in chapter 2 having in mind what was learned in chapter 3 to arrive to an optimal design.

During this chapter the hardware and software design will be explained in detail, with special focus, since the board ahead is mainly focused on digital electronics implementation using a microcontroller unit.

The aim of this part is to build a system that follows the block model presented below in figure 4.1.

### 4.1 Electronic design

In this section an in depth revision of the interconnection of the modules and modules itself. This device was created with the purpose of being a PCB that is able to support all the components required and at the same time interconnect them making it a fully functional circuit. For this project the board was designed using the EDA software Altium designer 2016, which offers a complete integration between the schematic design and the board design and routing, giving a option to 3D view the board itself. The design Will be kept as small as possible with only two layers top and bottom to be able to produce the PCB manually and kept it as *low-cost* as possible.

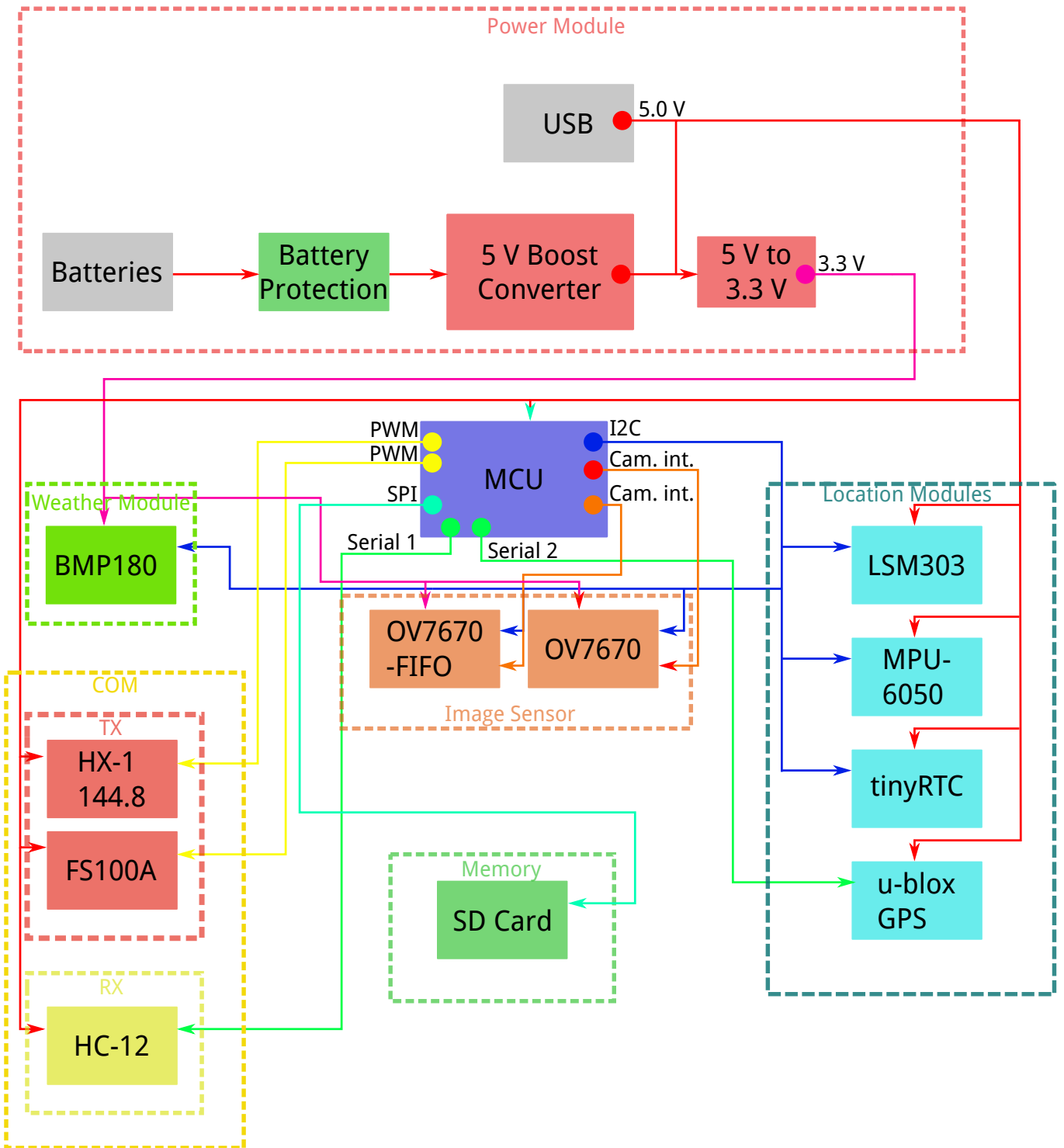


Figure 4.1 – Final block model of the design

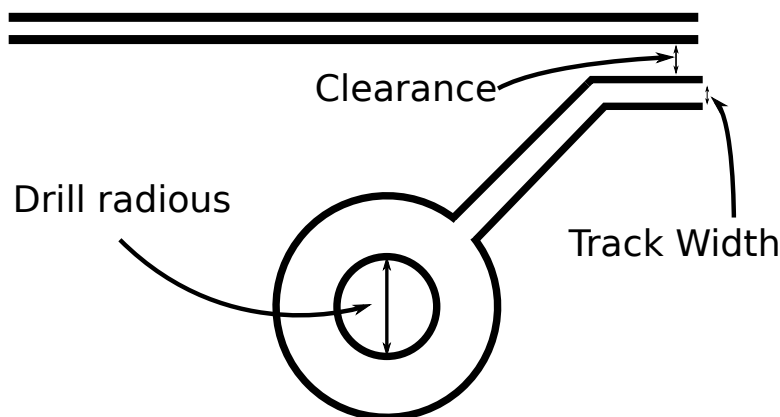
4

### 4.1.1 PCB design

This section introduces the final design of the full board and the design constraints that were dealt with. Each module interconnections will be explained in future sections with more detail. During the creation of this board we are confronted with many limitations due to the manufacturing process for the test board, that is home made through the chemical reaction  $\text{Cu} + \text{FeCl}_3 \rightarrow \text{CuCl}_2 + \text{Fe}$ , for that reason the main key design rules were obtained through trial and error. The final rules are displayed in table 4.1. In case of doubt of what the terms mean, an explanatory graph is displayed in figure 4.2.

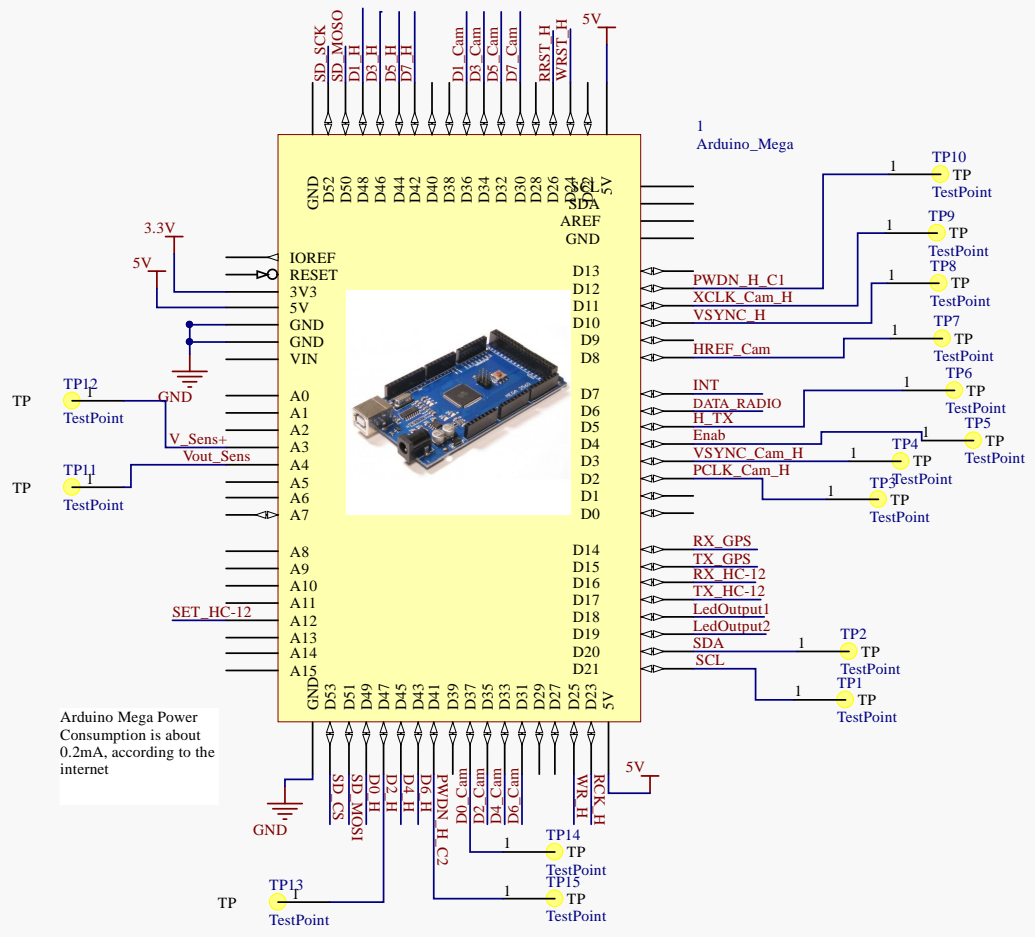
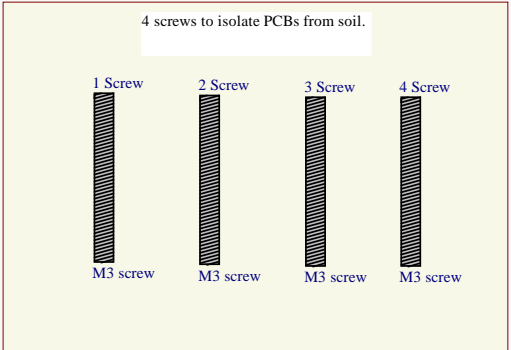
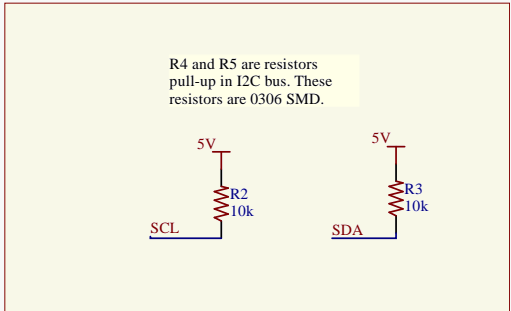
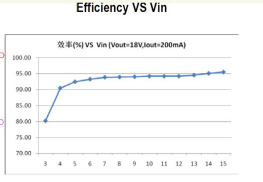
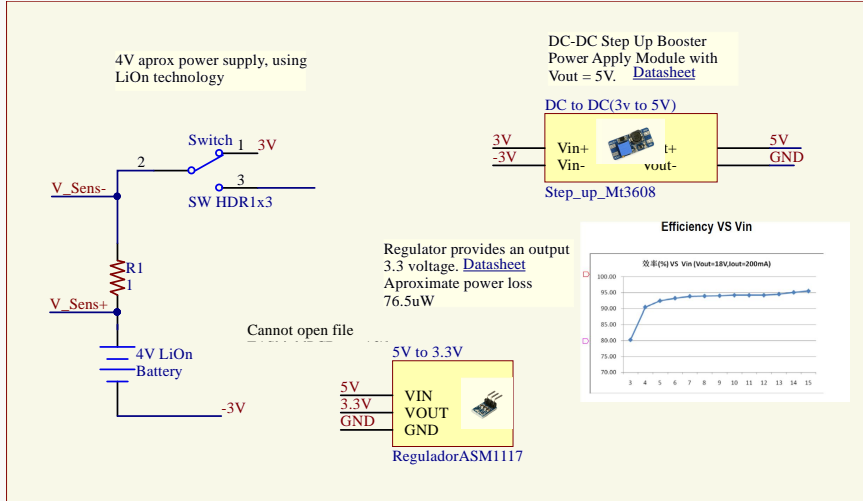
PCB design rules	
Track width	0.15 mm
Clearance	0.15 mm
Drill radius	0.15 mm

**Table 4.1** – Design rules for chemical reaction PCB manufacture



**Figure 4.2** – Explanation of PCB rules

Now the final schematics of the design will be displayed.

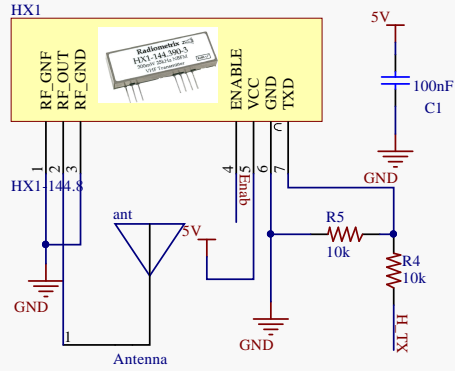


Designer's signature	Sheet title: <b>Microcontroler and Power Supply Schematics</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andrés Roldán Aranda
	Project title: <b>Trackuino_Shield.PrjPcb</b>		
Supervisor's signature	Desginer: <b>Luis Sánchez Velasco</b>		Sheet 1 of 8
	Date: <b>04/07/2017</b>	Revision: <b>04/07/2017</b>	



Additional pins to connect the radio module HX1-144.800.

Input Impedance = 100KOhm  
RF\_out = 50 Ohm



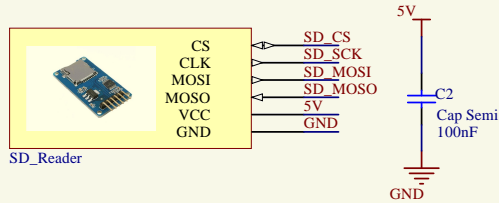
#### Technical Summary

- Transmit power: 300mW (24.7dBm)
- Operating frequency: 144.390, 144.800 and 169.4125MHz
- Channel spacing: 25kHz
- Supply: 5V (regulated)
- Current consumption: 140mA nominal transmit
- Data bit rate: 3kbps or 10kbps max.
- Size: 43 x 15 x 5mm

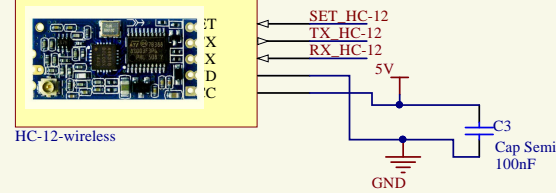
#### Baseband

Modulation bandwidth @ -3dB	6	0	-	5	kHz	6
TXD input level (logic low)	6	-	0	-	V	7
TXD input level (logic high)	6	-	5.0	-	V	7

A SD Card apadter transmitting over SPI containing a Chip Select feature



wirelessRS232



#### Product Features

- Long-distance wireless transmission (FU3: 1000m in open space, baud rate 5000bps in the air. FU4: 1800m in open space, baud rate 500bps in the air)
- Working frequency range (433.4-473.0MHz, with 100 communication channels)
- Maximum 100mW (20dBm) transmitting power (8 levels of power can be set)
- Four working modes, adapted to different application situations
- Built-in MCU performs communication with external device through serial port, no programming or configuration required for basic use
- The number of bytes transmitted continuously is unlimited (FU1 and FU3 modes only)
- Update software version through the serial port

Designer's signature

Sheet title: **Sensors Schematics Page 1**

Supervisor's signature

Project title: **Trackuino\_Shield.PrjPcb**

Desginer: **Luis Sánchez Velasco**

Date: **04/07/2017**

Revision: **04/07/2017**

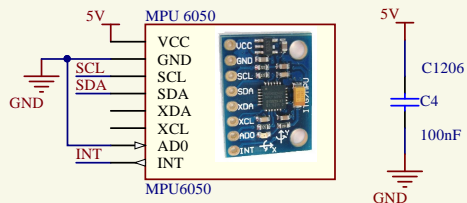
Sheet 2 of 8

Dpto. Electrónica y Tecnología  
de Computadores  
University of Granada  
C/ Fuente Nueva, s/n, 18001  
Granada, Granada, Spain  
Sr. Andrés Roldán Aranda





The MPU6050 sensor contains a MEMS accelerometer and a MEMS gyroscope in a single chip. It's connected to I2C bus.



- 5.2 Accelerometer Features**  
The triple-axis MEMS accelerometer in MPU60X0 includes a wide range of features:
- Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
  - Integrated 18-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
  - Accelerometer normal operating current: 500µA
  - Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
  - Orientation detection and signaling
  - Tap detection
  - User-programmable interrupts
  - High-G interrupt
  - User self-test

Provides an auxiliary I2C master if.  
I2C address = 0X68/69 the last address is modified by setting AD0 input (positive logic).  
Int pin provides an interrupt signal  
Most relevant registers:  
(16 bit) Accel\_X = 0x3B, 0x3C  
(16 bit) Accel\_Y = 0x3D, 0x3E  
(16 bit) Accel\_Z = 0x3F, 0x40  
(16 bit) Temp = 0x41, 0x42

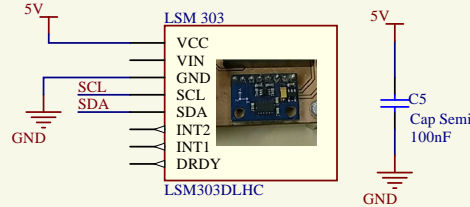
**Burst Read Sequence**

Master	S	AD+W	RA	S	AD+R	ACK	DATA	DATA	NACK	P
Slave		ACK	ACK		ACK					

**Burst Write Sequence**

Master	S	AD+W	RA	DATA	DATA	P
Slave		ACK	ACK	ACK	ACK	

The LSM303 combines a digital 3-axis accelerometer and 3-axis magnetometer into a single package that is ideal for making a tilt-compensated compass.



This device uses an I2C line to transfer data from the module to the MCU. These device is splitted to 2 interfaces accessed by the same input port:

Accelerometer I2C address = 0x19  
Magnetometer I2C address = 0x1E

Most relevant I2C registers:

Accelerometer:  
(16 bits) X\_accel = 0x28 , 0x29  
(16 bits) Y\_accel = 0x2A, 0x2B  
(16 bits) Z\_accel = 0x2C, 0x2D

Magnetometer:  
(16 bits) X\_magnetometer = 0x28 , 0x29

**Table 11. Transfer when master is writing one byte to slave**

Master	ST	SAD + W		SUB	DATA		SP
Slave			SAK		SAK		SAK

**Table 12. Transfer when master is writing multiple bytes to slave:**

Master	ST	SAD + W		SUB	DATA	DATA	SP
Slave			SAK		SAK	SAK	SAK

**Table 13. Transfer when master is receiving (reading) one byte of data from slave:**

Master	ST	SAD + W		SUB	SR	SAD + R		NMAK	SP
Slave			SAK		SAK		SAK	DATA	

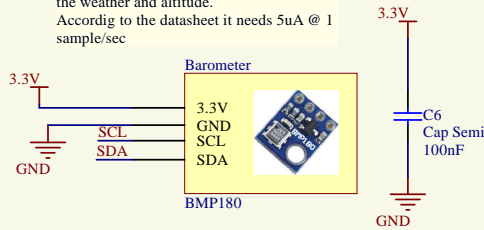
Idd	Current consumption in normal mode <sup>(2)</sup>		0.83	mA
-----	---	--	------	----

Designer's signature	Sheet title: <b>Sensor Schematics Page 2</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andrés Roldán Aranda
	Project title: <b>Trackuino_Shield.PrjPcb</b>		
Supervisor's signature	Desginer: <b>Luis Sanchez Velasco</b>		Date: <b>04/07/2017</b>   Revision: <b>04/07/2017</b>   Sheet 3 of 8



[Datasheet](#)

The BMP180 is barometric pressure sensor measure the absolute pressure of the air around him. This pressure varies with both the weather and altitude.  
According to the datasheet it needs 5uA @ 1 sample/sec



Register Name	Register Address
out_xlsb	F8h
out_lsb	F7h
out_msb	F6h
ctrl_meas	F4h
soft_reset	E0h
id	D0h
calib21 downto calib0	BFh down to AAh

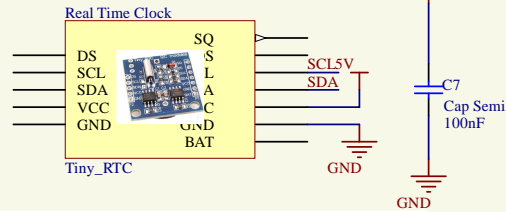
Write/Read I2C address = 0xEE / 0xEF

The device doesn't support burst reading.

Write:  
S - WADD - ACKS - REG - ACKS - DATAM - ACKS - P

Read:  
S - WADD - ACKS - REG - ACKS - P (restart) --  
S - RADD - ACKS - DATAS - ACKM - DATAM ----- DATAM - NACKM - S

Real Time Clock has dual ports I2C bus (SCL, SDA) and power (VCC, GND). We are only connecting to the RTC module DS1307. [Datasheet](#)



I2c write = 0xD0  
I2c read = 0xD1

Figure 4. Data Write—Slave Receiver Mode

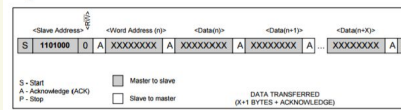


Figure 5. Data Read—Slave Transmitter Mode

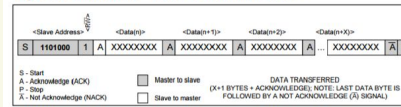
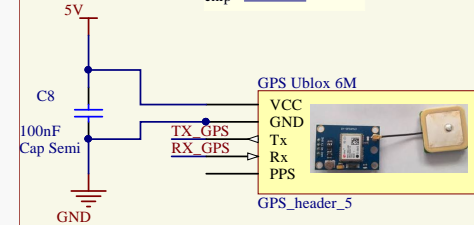


Table 2. Timekeeper Registers

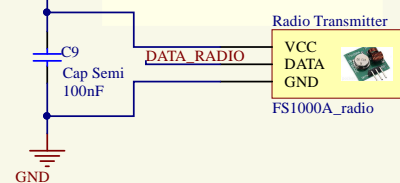
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH								10 Seconds	Seconds
01h	0								Seconds	Minutes
									Minutes	Minutes
02h	0								10 Minutes	Hours
									Hours	Hours
									Hours	1-12
									Hours	+AM/PM
									Hours	00-23
03h	0	0							10 Hour	DAY
									Day	Day
									Day	01-07
04h	0	0							10 Date	Date
									Date	Date
									Date	01-31
05h	0	0							10 Month	Month
									Month	Month
									Month	01-12
06h									10 Year	Year
									Year	Year
									Year	00-99
07h	OUT	0	0						Control	—
									RAM	—
									RAM	56 x 8
08h-3Fh									Control	00h-FFh

0 = Always reads back as 0.


GPS ublox NEO-6M, all calculations are done in chip [Datasheet](#)

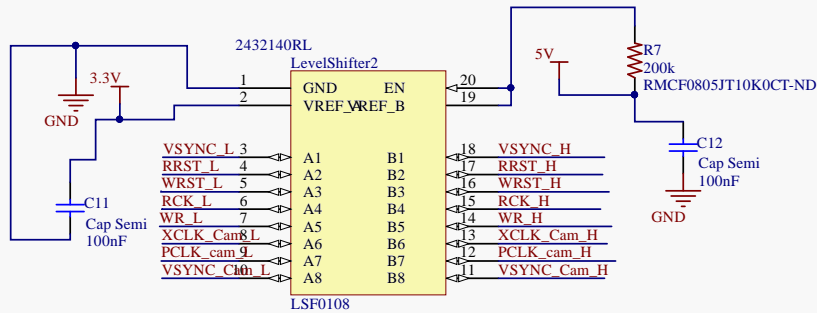
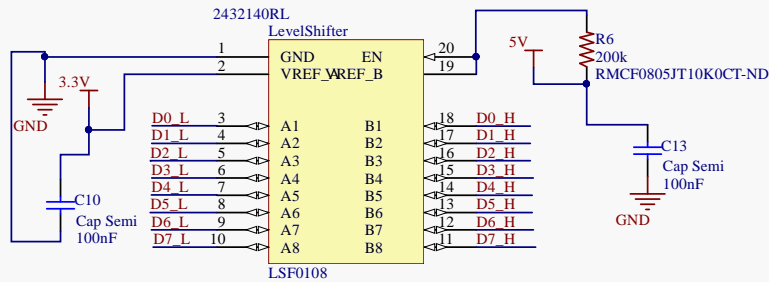


RF transmitter working frequency of 433Mhz.



The range in this transmitter is significantly lower than the hx1, also the bandwidth. This will be used as an off transmitter

Designer's signature	Sheet title: <b>Document Schematics Page 3</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andrés Roldán Aranda
	Project title: <b>Trackuino_Shield.PrjPcb</b>		
Supervisor's signature	Desginer: <b>Luis Sánchez Velaco</b>		
	Date: <b>04/07/2017</b>	Revision: <b>04/07/2017</b>	



The **LSF** family of devices consists of passive FET switches that use external pullup resistors to translate between voltage levels. The engineer controls the switching threshold of the device by changing  $V_{ref A}$ .  $V_{ref B}$  must be  $\geq 0.8$  V higher than  $V_{ref A}$  with a max  $V_{ref B}$  of 5 V, tied to EN (for **LSF010x** only), and pulled up to its supply voltage through a 200-k $\Omega$  resistor (for **LSF010x** only). The **LSF** devices are best suited for translating signals with fast edge rates, and perform well at frequencies up to 200 MHz (down translation) and 100 MHz (up translation). In some special down-translation applications, **LSF** devices do not require any pullups.

More information on the **LSF** family is found in "[Voltage-Level Translation with the LSF Family](#)," (SLVA675).

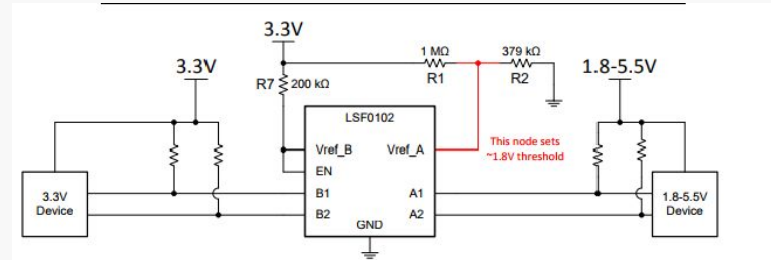



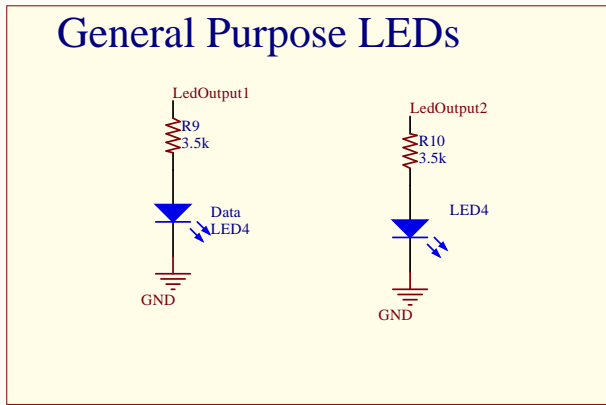
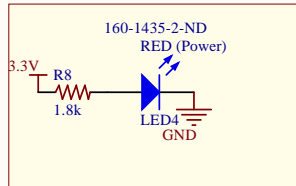
Figure 10. LSF0102 Single-Supply Operation Setup

PARAMETER	FROM (INPUT)	TO (OUTPUT)	$C_L = 50$ pF		$C_L = 30$ pF		$C_L = 15$ pF		UNIT
			TYP	MAX	TYP	MAX	TYP	MAX	
$t_{PLH}$	A or B	B or A	1.1	0.7	0.3			ns	
$t_{PHL}$			1.2	0.8	0.4				

Designer's signature	Sheet title: <b>Voltage Level Translator Array</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andrés Roldán Aranda	
	Project title: <b>Trackuino_Shield.PrjPcb</b>			
Supervisor's signature	Designer: <b>Luis Sanchez Velasco</b>		Sheet 5 of 8	
	Date: <b>04/07/2017</b>	Revision: <b>04/07/2017</b>		

Assuming 1.5V w/ 1mA drop in LED=>  
 $R = (3.3 - 1.5) \text{ V} / 1 \text{ mA} = 1.8 \text{ k}\Omega$

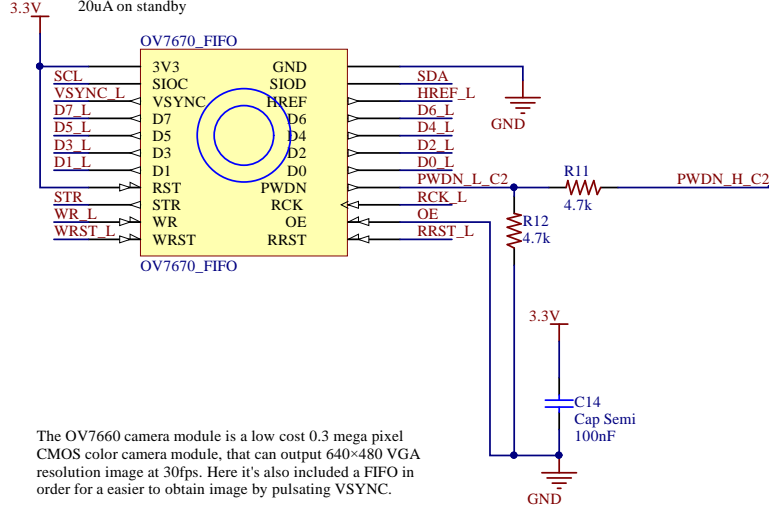
Assuming 1.5V w/ 1mA drop in LED=>  
 $R = (5 - 1.5) \text{ V} / 1 \text{ mA} = 3.5 \text{ k}\Omega$



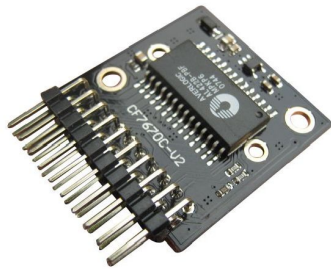
Designer's signature	Sheet title: <b>LEDs</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andrés Roldán Aranda
	Project title: <b>Trackuino_Shield.PrjPcb</b>		
Supervisor's signature	Desginer: <b>Luis Snánchez Velasco</b>		
	Date: <b>04/07/2017</b>	Revision: <b>04/07/2017</b>	

FIFO datasheet

According to the datasheet it needs less than 20uA on standby

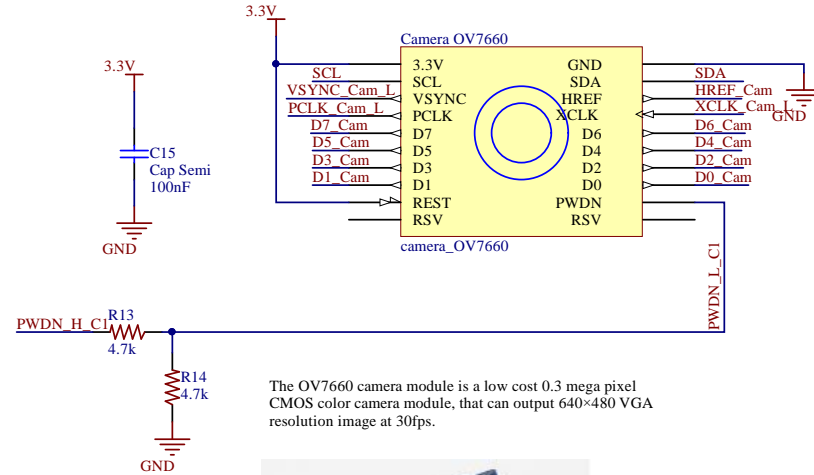


The OV7660 camera module is a low cost 0.3 mega pixel CMOS color camera module, that can output 640x480 VGA resolution image at 30fps. Here it's also included a FIFO in order for a easier to obtain image by pulsating VSYNC.



ov7670 datasheet

According to the datasheet it needs less than 20uA on standby

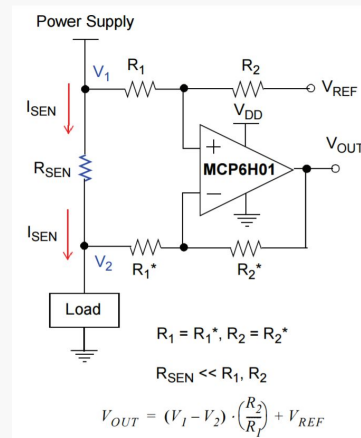
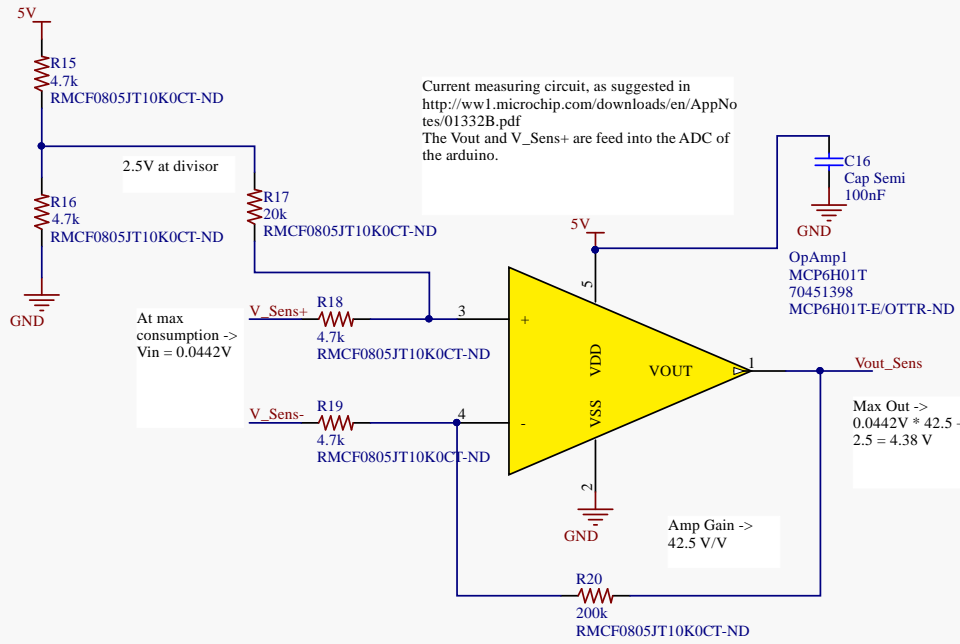


The OV7660 camera module is a low cost 0.3 mega pixel CMOS color camera module, that can output 640x480 VGA resolution image at 30fps.

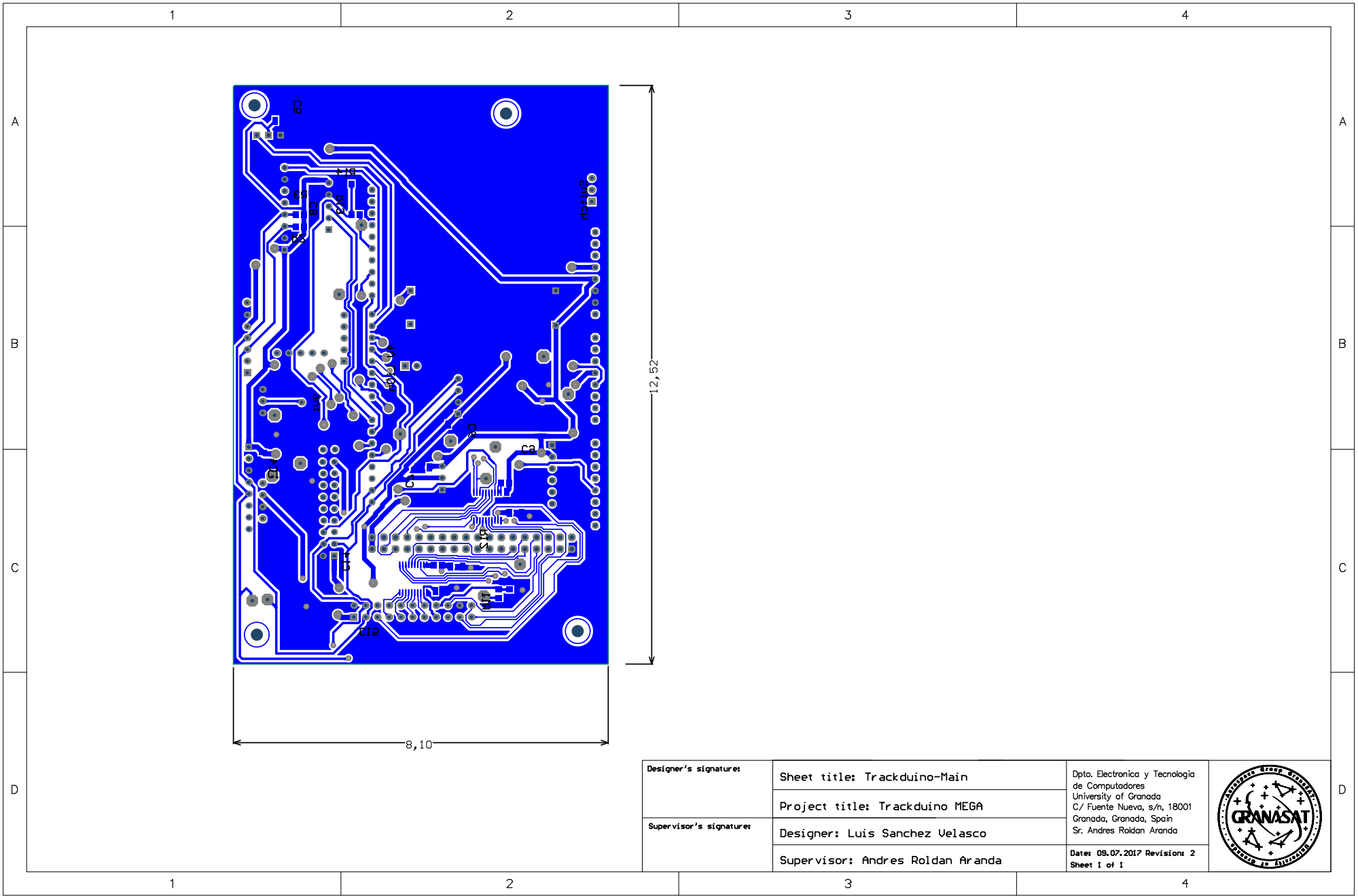


Another camera, just because, never enough...

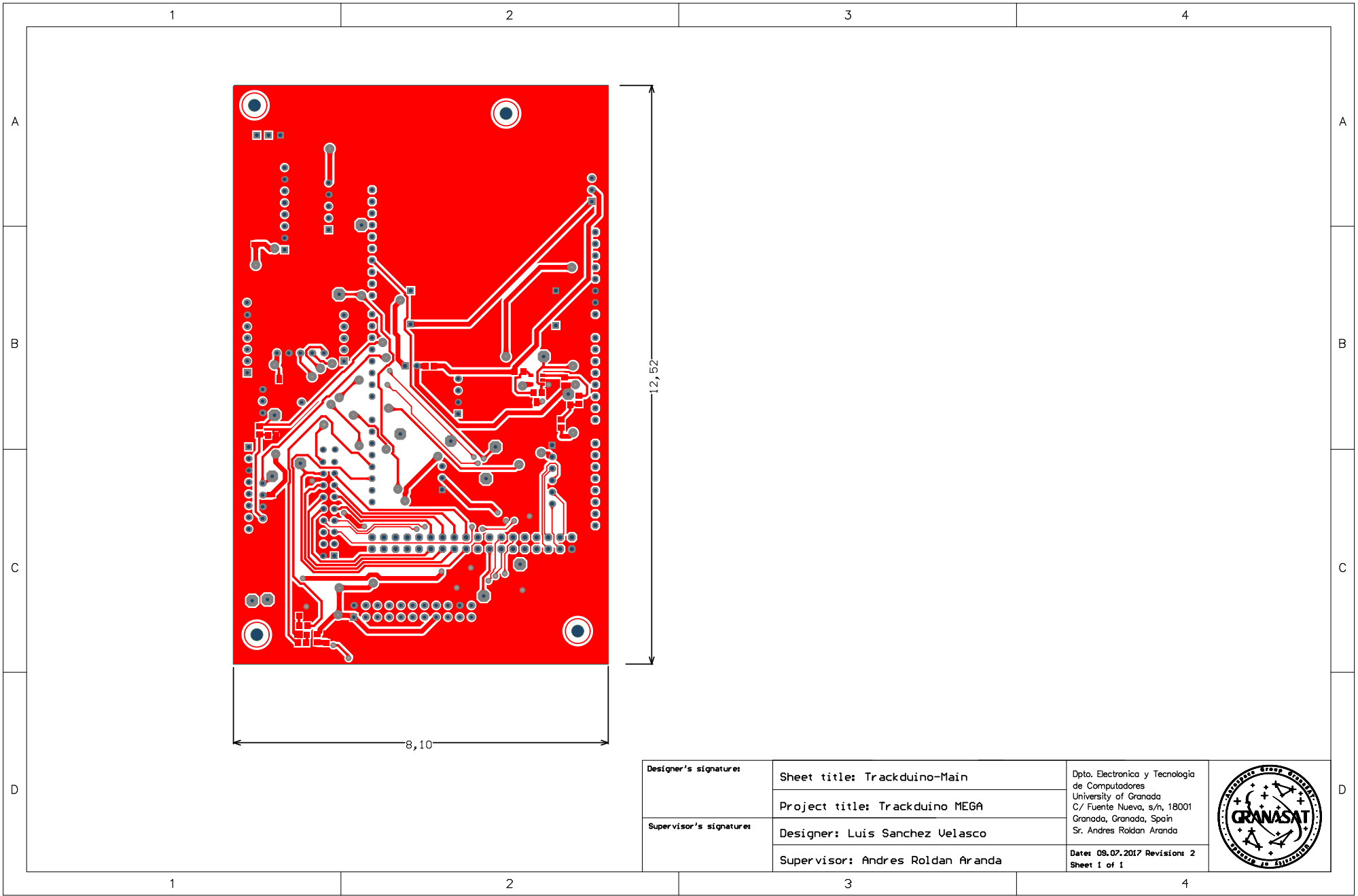
Designer's signature	Sheet title: <b>Cameras</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andrés Roldán Aranda
	Project title: <b>Trackuino_Shield.PrjPcb</b>		
Supervisor's signature	Desginer: <b>Luis Sánchez Velasco</b>		
	Date: <b>04/07/2017</b>	Revision: <b>04/07/2017</b>	



Designer's signature	Sheet title: <b>Current Measuring System</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andrés Roldán Aranda
	Project title: <b>Trackuino_Shield.PrjPcb</b>		
Supervisor's signature	Desginer: <b>Luis Sánchez</b>		
	Date: <b>04/07/2017</b>	Revision: <b>04/07/2017</b>	

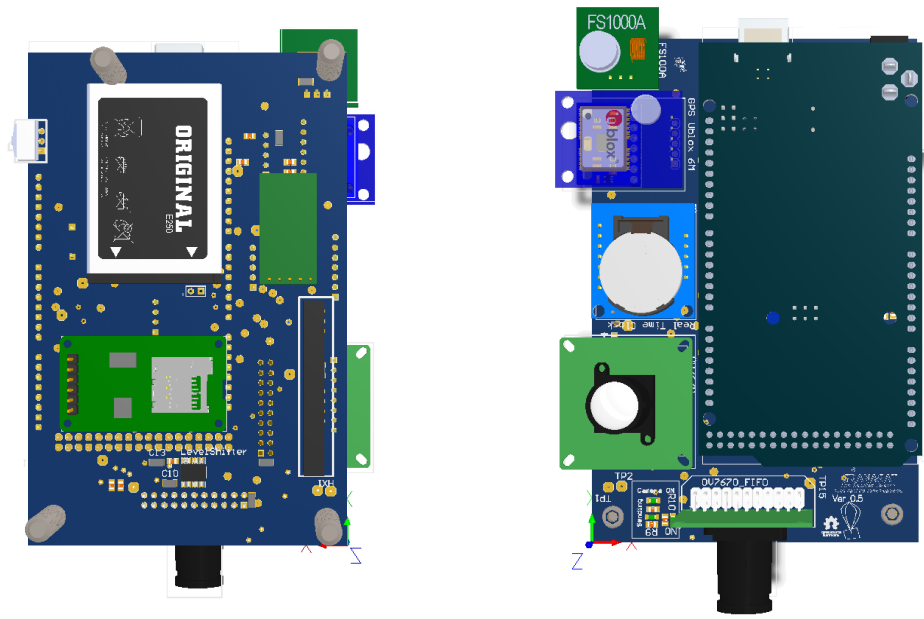


Designer's signature:	Sheet title: Trackduino-Main	Dpto. Electronica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda	
	Project title: Trackduino MEGA		
Supervisor's signature:	Designer: Luis Sanchez Velasco	Date: 09.07.2017 Revisión: 2 Sheet 1 of 1	
	Supervisor: Andres Roldan Aranda		



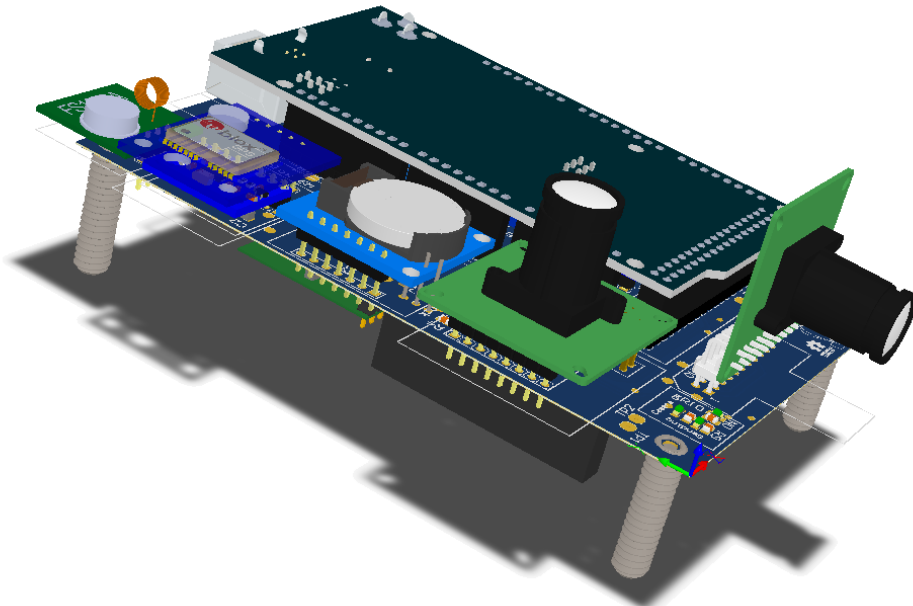
Designer's signature:	Sheet title: Trackduino-Main	Dpto. Electronica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda	
	Project title: Trackduino MEGA		
Supervisor's signature:	Designer: Luis Sanchez Velasco	Date: 09.07.2017 Revisión: 2 Sheet 1 of 1	
	Supervisor: Andres Roldan Aranda		





(a) Zero bottom view

(b) Zero top view



(c) Orthogonal view

Figure 4.3 – PCB's 3D model

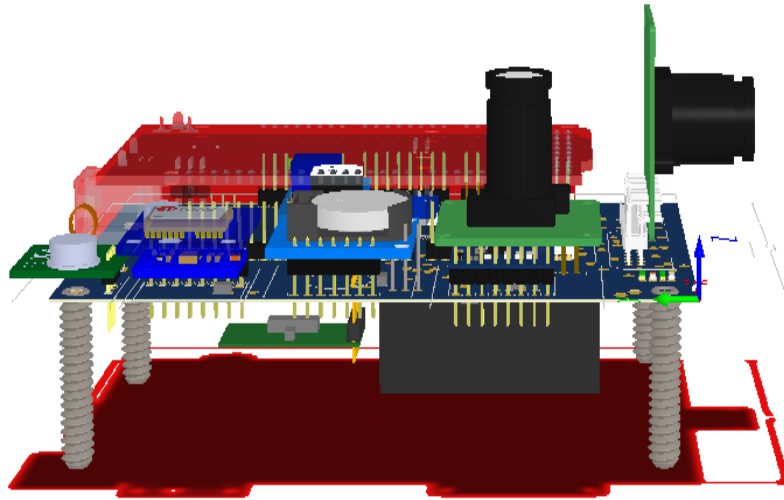


Figure 4.4 – Video of the PCB

#### 4.1.1.1 Power budget

To fully understand the power drawn by the components a more advanced power budget is needed, having in mind the frequency of usage of the different components, as well as the converter's efficiency.

From the last table we can calculate the overall efficiency having in mind that the average current consumption is going to be of about 200 mA, and that 2.01 mA are going to be translated to 3.3 V. By looking at the MT3608's datasheet [32], the switched regulated that will be used, it can be read that the efficiency will be of about 91%. At the 5V to 3.3V linear regulator (assuming the simple model) will be of  $(5 - 3.3) \cdot 2.01 \text{ mW} = 3.57 \text{ mW}$ . On average the total power drawn from the battery will be:

$$\frac{938 + 3.57}{0.91} = 1034.69 \text{ mW} \rightarrow \frac{1034 \text{ mW}}{3.7 \text{ V}} = 280 \text{ mA} \quad (4.1.1)$$

#### 4.1.1.2 LSF0108 voltage level translation

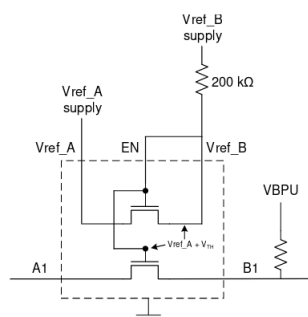
The voltage translation made by the integrated circuit LSF0108 might be tricky to understand. That section is written to clear any doubts of how the implementation is

Component	Active consumption (mA)	Power down consumption ( $\mu A$ )	Active time (ms)	Frequency (Hz)	Consumption per cycle (mA/Hz)	Voltage (V)	Power (mW)
BMP180 [26]	1	$\approx 0$	$2 \times 5$	1	$10 \cdot 10^{-3}$	3.3	$33 \cdot 10^{-3}$
LSM303 [30]	0.110	1	5	-	0.110	5	0.55
NEO-6M [11]	47	-	-	1	47	5	235
DS1307 [31]	1.5	200	$250 \cdot 10^{-3}$	1	0.25	5	1.25
$2 \times$ OV7670 [12]	10	100	500	60	1.075	3.3	3.35
HX1 [13]	140	-	-	-	140	5	700
FS1000A [29]	10	-	-	-	10	5	50
HC-12 [15]	200	-	-	-	200	5	1000
Arduino	25	-	-	-	25	5	125
Total	469	-	-	-	213 <sup>a</sup>	-	938

Table 4.2 – Device's power budget

<sup>a</sup>Excluded FS1000A and HC-12 modules since they are going to be rarely used

made. For an application of such IC the application notes on [33] will be looked at. From the previous reference it can be seen that for a proper functioning,  $V_{ref} A$  should have the lower voltage, meanwhile  $V_{ref} B$  should have the largest one, and be connected to the source by a  $200k\Omega$  resistor, as depicted in figure 4.5.



**Figure 4.5** – *LSF basic setup*

As for the pull-up resistor, it can be read at the manual that they are only needed in situations when the driving signal is an open drain source (e.g. I2C) or when the A port is translating to a voltage that is not equal to  $V_{ref} A$ . For side B they are only needed when it is used as an output. In our case they will not be needed since their application is to drop the voltage from 5V to 3.3V for usage in the OV7670 cameras. It can be seen in figure 4.6 that this is not the case for the first LSF0108 used in the schematic, whose mission is to bring up the voltage from the camera image output register to 5V, for that particular case the internal pull-up of the ATmega2560 can be used.

## 4.2 Software design

For this section, the popular platform [Arduino](#) will be used to flash the [MCU](#). The software will try to be as modular as possible, as well as creating a class system that makes sense with the whole application. During this section we will describe and explain the classes for each module to finally create a logic that is able to control each class and trigger the functions when the time is required, therefore a bottom-top design approach will be used.

### 4.2.1 Activity diagram

Activity diagrams are really useful to decompose a complex process into several small sequential actions, for this particular device the actions will be taking considering that no external agent communicates with the [PCB](#), meaning that the work flow will not change based on an administrator. Below the activity diagram for the device is introduced in figure 4.7.

The device will firstly initialize all the corresponding modules by writing the needed registers and the starting the clock sources. After all that is done a new frame to be sent

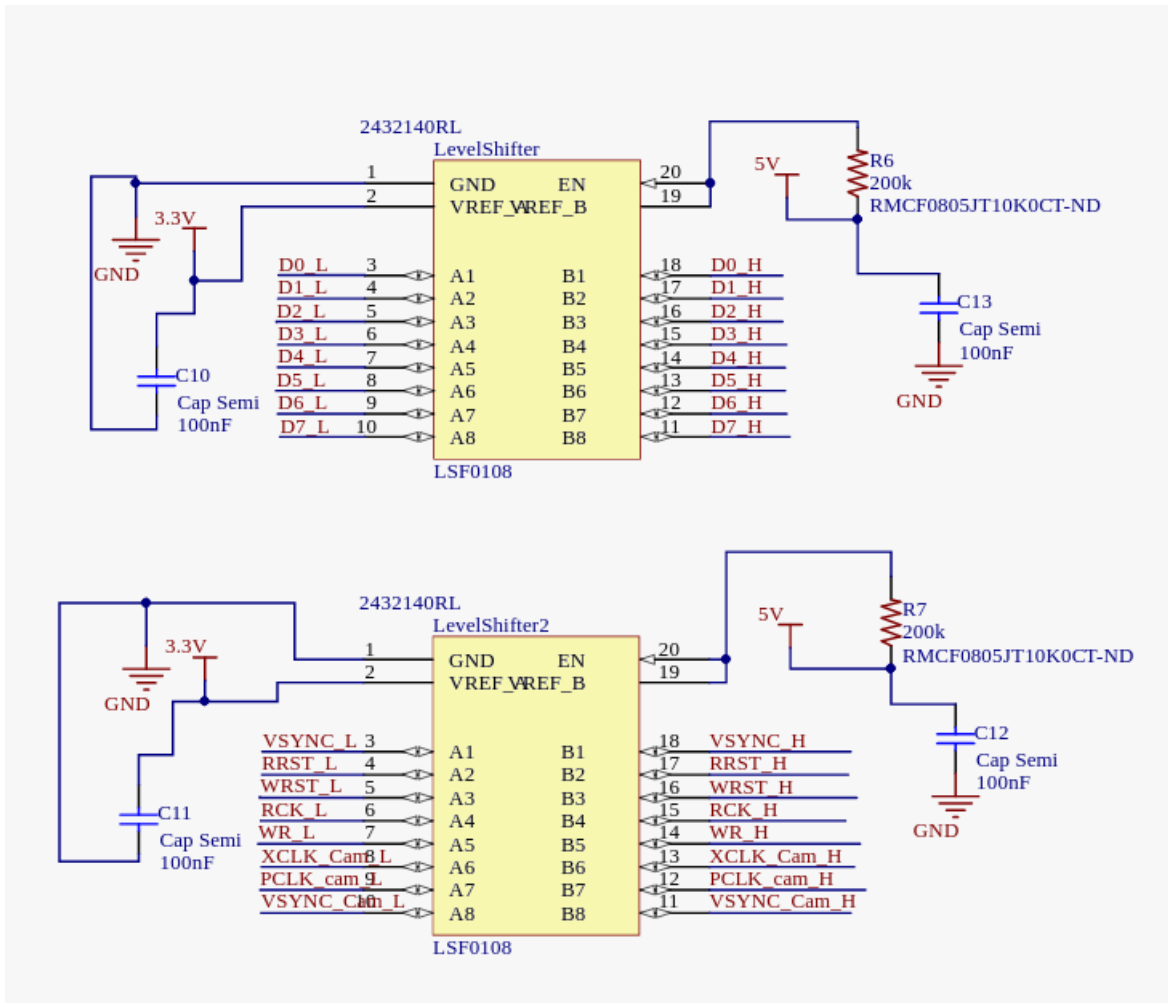


Figure 4.6 – LSF0108 usage in the schematics

will be created and flushed, below it is shown the activity diagram for the frame generation in figure 4.8.

As it can be seen, the **MCU** checks if every component is still functioning properly and then reads the corresponding data from it, this data is stored in a log file in the SD card. The cameras work in the same fashion, but storing the data as soon as it is picked up, due to memory considerations. This data is then saved in a buffer to be later flushed by the activity shown below in figure 4.9

This last part will take the data from the buffer and transform it into an **AFSK** signal, that is then going to be sent out as a **PWM** signal to the transmission module.

#### 4.2.2 BMP180

This little module is very easily interfaced with the rest of the board, just connecting the **I2C** with the board and creating the power lines will be enough to make it work. In order

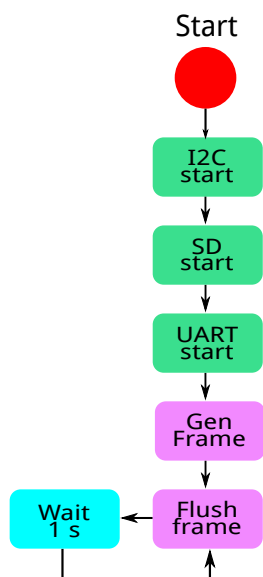


Figure 4.7 – Main activity diagram

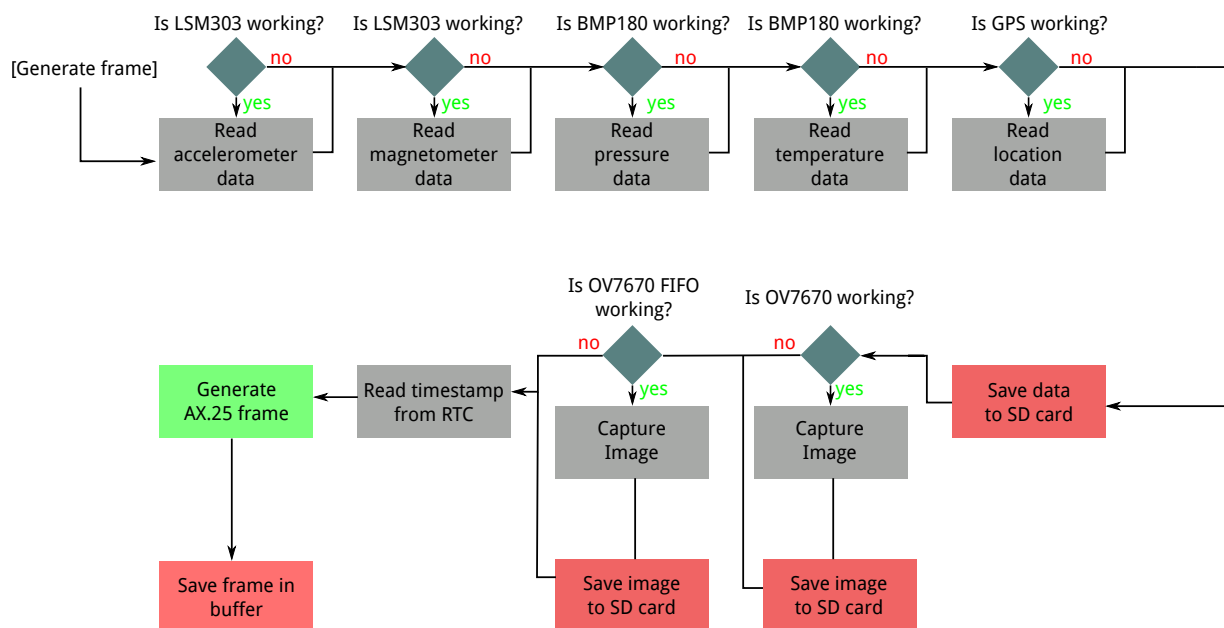


Figure 4.8 – Frame creation diagram

to obtain data from this device we will use the [I2C](#) protocol, for that we need the address that is 0x77, and the respective memory map for the device is shown at [table 4.3](#).

For this module to work properly we need to set the proper number in the Control Measurement register, by doing this we can obtain a single measurement of either temperature or pressure. This data can be obtained by reading the Output LSB and the Output MSB. The temperature is calculated by reading firstly the calibration registers and

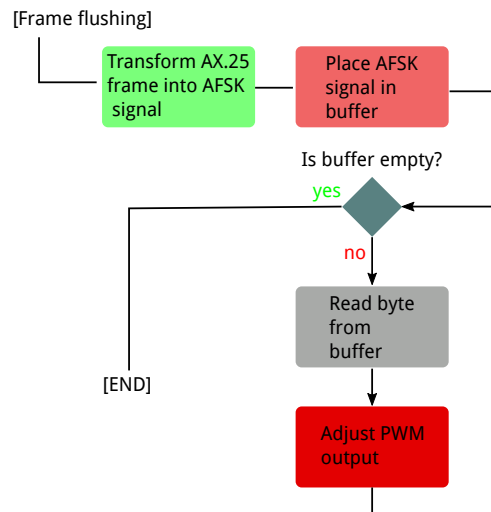


Figure 4.9 – Frame flushing diagram

I2C Address	0x77									
Register Name	Register address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
OUT XLSB	0xF8	ADC output 7:3					0	0	0	0x00
OUT LSB	0xF7	ADC output lsb								
OUT MSB	0xF6	ADC output msb								
CTRL MEAS	0xF4	OSS 1:0		SC0	Measurement control				0x00	
RESET	0xF6	RESET								0x00
ID	0xD0	ID								0x55
Calibration	0xD0	Calibration, read only								n/a

Table 4.3 – Register mapping for the BMP180

then following the equations provided in [34].

$$t_u = 256 \cdot MSB + LSB \quad (4.2.1)$$

$$\alpha = c_5(t_u - c_6) \quad (4.2.2)$$

$$T = \alpha + \frac{m_c}{m_d + \alpha} \quad (4.2.3)$$

```

1 char BMP180::getTemperature(double &T)
2 {
3     unsigned char data[2];
4     char result;
5     double tu, a;
6
7     data[0] = BMP180_REG_RESULT;
8
9     result = readBytes(BMP_I2C_ADRESS, data, 2);
10    if (result) // good read, calculate temperature
11    {
12        tu = (data[0] * 256.0) + data[1];
13        a = c5 * (tu - c6);
14        T = a + (mc / (a + md));
15    }
16 }
  
```

```

17 |         return(result);
18 |     }

```

The pressure calculation is far more complex involving more constants and also the previous obtained temperature. The following code will allow us to obtain the pressure value:

```

1 | char BMP180::getPressure(double &P, double &T)
2 | {
3 |     unsigned char data[3];
4 |     char result;
5 |     double pu,s,x,y,z;
6 |
7 |     data[0] = BMP180_REG_RESULT;
8 |
9 |     result = readBytes(BMP_I2C_ADRESS ,data, 3);
10 |    if (result) // good read, calculate pressure
11 |    {
12 |        pu = (data[0] * 256.0) + data[1] + (data[2]/256.0);
13 |
14 |        s = T - 25.0;
15 |        x = (x2 * pow(s,2)) + (x1 * s) + x0;
16 |        y = (y2 * pow(s,2)) + (y1 * s) + y0;
17 |        z = (pu - x) / y;
18 |        P = (p2 * pow(z,2)) + (p1 * z) + p0;
19 |
20 |    }
21 |    return(result);
22 | }

```

### 4.2.3 LSM303

This sensor is as the previous one, directly interface by the I2C protocol. It provides two different addresses for communicating with the accelerometer or the magnetometer. Both gain and range can be calibrated [9]. The relevant registers for this devices are listed in table 4.4.

I2C Address	Accelerometer :0x19				Magnetometer :0x1E					
Register Name	Register Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
CTRL REG1 ACCEL	0x20	Data rate 3:0				LP	Zen	Yen	Xen	0x07
OUT X LSB ACCEL - OUT Z MSB ACCEL	0x03-0x08	Output data								0x00
MAG MR REG	0x02	0					MD 1:0		0x00	
MAG CRB REG	0x01	Gain selection 2:1			0			0x00		
OUT X MSB MAG - OUT Z LSB MAG	0x03-0x08	Output data								0x00

Table 4.4 – Register mapping for the LSM303

In order to obtain data from the accelerometer it is first needed to be initialized, this can be achieved by writing a non zero value in the data rate bits, located in the Control Register 1 of the accelerometer, this value will change the acquisition frequency. The data can be



retrieved then by reading the output register and dividing then by the scale factor, as seen in this code snippet.

```

1 | char LSM303::getAccelerometer(S3DVEC &accel){
2 |     char data[6];
3 |     char ok;
4 |     data[0] = LSM303_REGISTER_ACCEL_OUT_X_L_A | 0x80;
5 |     ok = readBytes(LSM303_I2C_ADDRES_ACCEL, data, 6);
6 |     accel.x = (double)(((data[1] << 8) | data[0]) >> 4)*
7 |         _lsm303Accel_MG_LSB * SENSORS_GRAVITY_STANDARD;
8 |     accel.y = (double)(((data[3] << 8) | data[2]) >> 4)*
9 |         _lsm303Accel_MG_LSB * SENSORS_GRAVITY_STANDARD;
10 |    accel.z = (double)(((data[5] << 8) | data[4]) >> 4)*
11 |        _lsm303Accel_MG_LSB * SENSORS_GRAVITY_STANDARD;
12 |    return(ok);
13 | }

```

The magnetometer, on the other hand, can work in two different modes, one in which we set the acquisition frequency and another one in which data is taken on demand. This behavior can be set by modifying bits MD in the Magnetometer MR Register. Gain and therefore range and scale is set in the first three bits in the CRA register. The data is obtained again by reading this register and multiplying by the scale. This is displayed in the following code snippet.

```

1 | char LSM303::get Magnetometer(S3DVEC &magn){
2 |     char data[6];
3 |     char ok;
4 |     data[0] = LSM303_REGISTER_MAG_OUT_X_H_M | 0x80;
5 |     ok = readBytes(LSM303_I2C_ADDRES_MAGNE, data, 6);
6 |     magn.x = (double)((data[0] << 8) | data[1]) /
7 |         _lsm303Mag_Gauss_LSB_XY;
8 |     magn.y = (double)((data[2] << 8) | data[3]) /
9 |         _lsm303Mag_Gauss_LSB_XY;
10 |    magn.z = (double)((data[4] << 8) | data[5] ) /
11 |        _lsm303Mag_Gauss_LSB_Z;
12 |    return(ok);
13 | }

```

#### 4.2.4 MPU-6050

This module is again interfaced by I2C wires. the I2C address of this device can be selected by setting high or low the pin A0, which is very handy since this module will share the same address than the RTC chip in the future. An accelerometer and a gyroscope can be accessed through this device. During this product the gyroscope will be the only sensor accessed since the motion data is already accessed with the accelerometer in the LSM303. The most relevant registers for this device basic usage are listed in the table 4.5 below:

The device will be used in low power mode due to the PCB nature, thus for setting up the

I2C Address	0x68 / 0x69									
Register Name	Register Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
PWR MGMT 1	0x6B	Reset	Sleep	Cycle	-	TEMP DIS	Clock sel 2:0			0x00
PWR MGMT 2	0x6C	LP wake		STB XA	STB YA	STB ZA	STB XG	STB YG	STB ZG	0x00
GYRO CONFIG	0x27	XG ST	YG ST	ZT ST	FS SEL 0:1			-		0x00
OUT X GYRO MSB - OUT Z LSB GYRO	0x43-0x67	Output data								0x00

**Table 4.5** – Register mapping for the MPU-6050

device we will start by writing a 1 in the Cycle bit in the first power management register, this will place the device in low power mode, where it will be put to sleep and woken up depending on the frequency selected in the LP wake bits, located on the second power management register. Once this is done we can select the Full Scale of the gyroscope by writing in the configuration register. This will produce a different scaling factor, the configuration of the device's scale is shown below.

```

1 char MPU6050::setRange(uint8_t fs){
2     uint8_t data[2];
3     data[0] = MPU6050_RA_GYRO_CONFIG;
4     switch (fs) {
5         case 0:
6             data[1] = MPU6050_GYRO_FS_250;
7             _scale = (float) 131.0;
8             break;
9         case 1:
10            data[1] = MPU6050_GYRO_FS_500;
11            _scale = (float) 65.5;
12            break;
13         case 2:
14            data[1] = MPU6050_GYRO_FS_1000;
15            _scale = (float) 32.8;
16            break;
17         case 3:
18            data[1] = MPU6050_GYRO_FS_2000;
19            _scale = (float) 16.4;
20            break;
21     }
22     return(writeBytes(MPU6050_DEFAULT_ADDRESS, data, 2));
23 }
```

#### 4.2.5 TinyRTC / DS1307

This device provides us with an external clock, really useful to produce time stamps to send alongside with the AX.25 messages. This module will be connected as well to the I2C bus. For reading and writing dates at this device we just need to take a look at the register mapping, depicted in table 4.6.

The numbers are kept in BCD inside the device's memory, therefore tools for making the translation between BCD and hexadecimal are needed. This last problem is easy to achieve if we assume that no overflow will occur since we are using regular dates:

```

1 | inline uint8_t decToBcd(uint8_t val)
```

I2C Address	0x68							
Register address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	CH	10 seconds			seconds			
0x01	0	10 minutes			minutes			
0x02	0	MODE	10 hour		hour			
0x03	0					WEEK DAY		
0x04	0	0	10 date		date			
0x05	0			10 month	month			
0x06	10 year				year			

**Table 4.6** – Register mapping for the DS1307

```

2 | {
3 |     return ( (val/10 << 4) + (val%10) );
4 | }
5 |
6 | inline uint8_t bcdToDec(uint8_t val)
7 | {
8 |     return ( ( (val >> 16)*10 ) + (val%16) );
9 | }

```

## 4

Once this problem is solved it just needed to read from registers 0x00 to 0x06 and save the information in a structure just like this:

```

1 | typedef struct TIME_FORMAT_t {
2 |     uint8_t second;           // 0-59
3 |     uint8_t minute;         // 0-59
4 |     uint8_t hour;           // 1-23
5 |     uint8_t dayOfWeek;      // 1-7
6 |     uint8_t dayOfMonth;     // 1-28/29/30/31
7 |     uint8_t month;          // 1-12
8 |     uint8_t year;
9 | } TIME_FORMAT;

```

### 4.2.6 GPS

The in board [GPS](#) module can be accessed through it serial interface, at 9600 baud, in case concerning us, it is located in the 3rd serial interface of the [Arduino](#) MEGA board. Through this serial interface the [MCU](#) will receive NMEA sentences, each of them representing location data in several ways, for example, time, altitude or route data. For our application an external library, provided in the original Trackduino source, is used due to the complexity of string processing in C. Only GGA and RMC sentences will be processed wince they will provide us with relevant data for our application.

- GGA - Essential fix data that provides a 3D location of the device. Altitude and time reference will be obtained from this sentence.

- RMC - Essential position, velocity and time.

As an example for one of this sentences we can see [35]:

GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W\*6A Where the important data is:

- GPRMC: This is a GPRMC sentence.
- 123519: Time, 12:35:19 UTC.
- 4807.038,N: Latitude 48 deg 07.038' N
- 01131.000,E: Longitude 11 deg 31.000' E
- \*6A: Checksum

#### 4.2.7 OV7670 - NO FIFO

This is one of the device's camera, this module is configured by its I2C interface. In order to be able to read the image pixels the sequence of binary outputs will go as follows, depicted in figure 4.10:

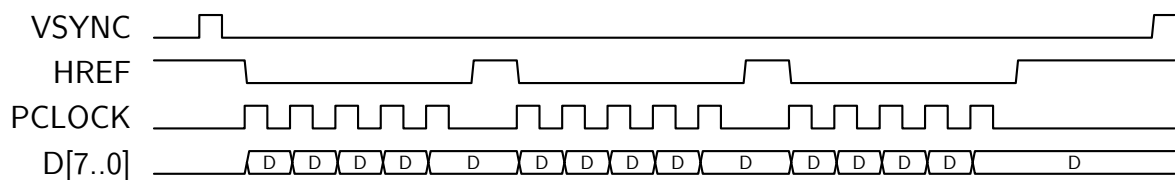


Figure 4.10 – Timing diagram of the OV7670's output

- VSYNC does a pulse, that means that the frame capture shall begin.
- PCLK goes high, that means a new byte has arrived.
- PCLK goes low.
- HREF will go high when the data is not valid, horizontal blanking.

Due to the MCU's constraints, it will not be able to record a full VGA image, that's why a QQVGA (160 × 120) image will be obtained instead.

### 4.2.7.1 Setup

In order to configure the camera we again need to take a look at the internal configuration layer and how are they mapped, however this could take a ridiculous amount of time, due to the complexity of the device, so for the configuration the default configuration available from the Linux OV7670 driver will be used [36].

An additional needs to be taken before, since the camera need an external clock source of at least 8 MHz that will be provided by the [Arduino](#) itself. In order to achieve this an internal PWM is going to be tweaked. Below is displayed the internals of an ATMEga 2560 in figure 4.11.

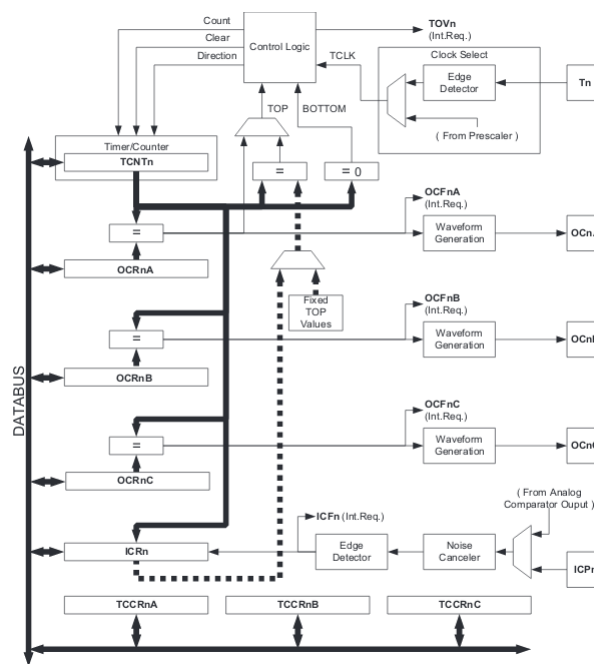


Figure 4.11 – 16 bit Timer/Counter of an ATMEga 2560 [7]

As it can be seen in the previous figure 4.11 this can be achieved the OCR1A register to 0x00 and to reset once the counter has reached the OCR1A value. This will make a match to happen every clock cycle, the only configuration left is to toggle the output every time a match occur, creating an output clock of  $\frac{16 \text{ MHz}}{2} = 8 \text{ MHz}$ .

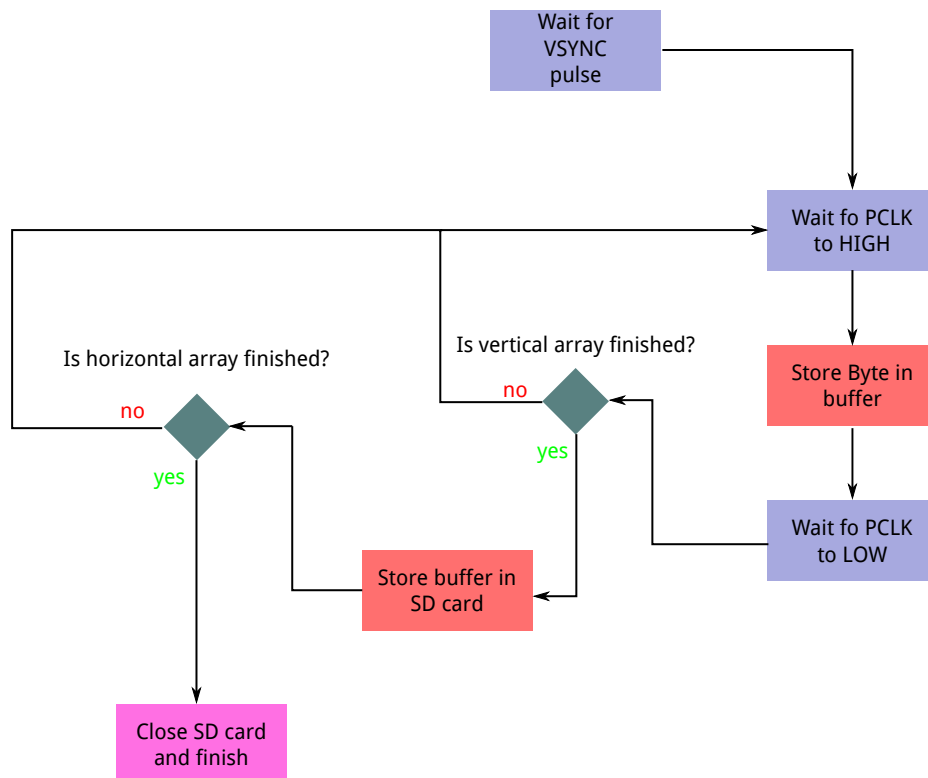
```

1 //setup 8MHz clock on pin 11
2 DDRB = (1 << 5);
3 OCR1A = 0x0; //top
4 TCCR1A = (1 << 6); // toggle on compare match
5 TCCR1B = (1 << 0) |(1 << 3); //no prescaling

```

### 4.2.7.2 Data collection

The main strategy to read a picture from this figure is to wait for a VSYNC pulse to occur, that means that a picture is taking place and new data will be available at D[7..0] at every pulse of PCLK. The main algorithm for collecting data is described below in figure 4.12.



**Figure 4.12** – *Picture extraction of the OV7670*

The main idea of the algorithm is to wait for the blanking time to occur, that is the time happening when the camera is still measuring data from the pixel array from outside the range of QQVGA, to write the image to the SD card.

It also sensible to note that a new byte event will occur at 8 MHz, meanwhile our MCU's frequency is of about 16 MHz, thus leaving no more than one machine instruction to place the read data on to a buffer, leaving us with need to program by writing on the registers directly, without using the Arduino library. Below is displayed the C++ function that collects an image and place on to the SD card.

```

1  static void captureImg(uint16_t wg,uint16_t hg){
2    uint16_t lg2;
3    uint8_t buf[320];
4    File dataFile = SD.open("datalog.txt", FILE_WRITE);
5    //Wait for vsync it is on pin 3 (counting from 0) portD
6    while(!(PINE& (1<<5)    ));//wait for high
  
```

```

7   while((PINE& (1 << 5)  ));//wait for low
8   /* This code is very similar to quga sending code except we
9   have even more blanking time to take advantage of */
10  while(hg--){
11    uint8_t*b=buf ,*b2=buf;
12    lg2=wg;
13    while(lg2--){
14      while((PINE&(1<< 4)));//wait for low
15      *b++= PINC;
16      while(!(PINE&(1 << 4)));//wait for high
17    }
18    /* Finish sending the remainder during blanking */
19    lg2=wg;
20    while(!( UCSROA & (1<<UDREO)));//wait for byte to transmit
21    while(lg2--){
22      dataFile.write(*b2++);
23      while(!( UCSROA & (1<<UDREO)));//wait for byte to transmit
24    }
25  }
26  dataFile.close();
27 }

```

## 4

This last function will grab a [QQVGA](#) image with YUV422 coloration, and place it on to the SD card. The memory used by this image will be:

$$160 \times 120 \times 2 = 38400 \text{ BYTES} \rightarrow 35.5\text{kB} \quad (4.2.4)$$

Note that the total area of the picture must be multiplied by 2 since YUV422 uses 2 bytes per pixel.

This image can be converted into an actual image format, such as PNG by using the open source library FFMPEG with the following command.

```

1 | ffmpeg -f rawvideo -s 160x120 -pix_fmt yuyv422 -i INPUTFILE.YUV -
2 | f image2 -vcodec
   | png OUTPUTFILE.PNG

```

#### 4.2.8 OV7670-FIFO

This is another version of the OV7670 camera, it consist of the same image sensor but instead of being directly interfaced to the [Arduino](#) itself it will be interfaced to an AL422VB FIFO chip as described in the picture below in figure [4.13](#) .

This connection will remove any time constraints that the board could have, since they will be solved by the FIFO, which can be read at any given speed without mayor problems.

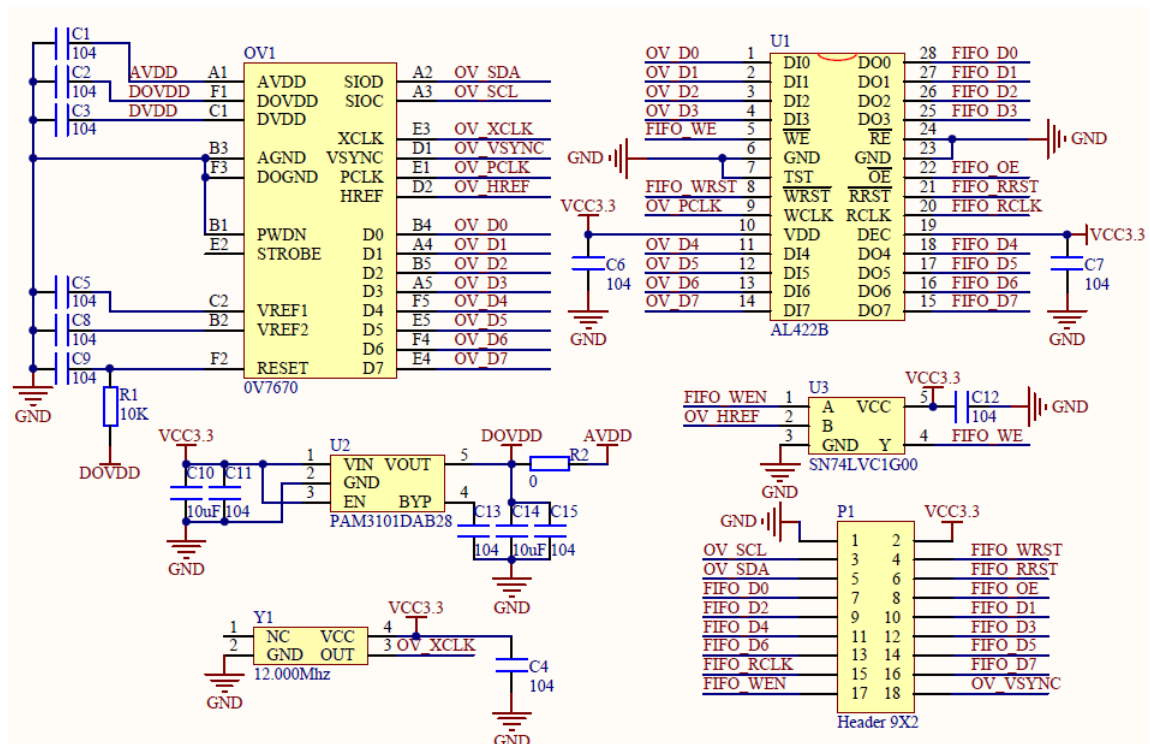


Figure 4.13 – OV7670 + FIFO schematics [8]

#### 4.2.8.1 Setup

The setup of this camera does not differ at all with the previously explained in section 4.2.7.1. The only difference is that VGA resolution will be used, since time constraints are removed.

#### 4.2.8.2 Data collection

As it can be seen from figure 4.13 and what was explained in the section 4.2.7 the following steps shall be taken in order to write an image from the camera to the FIFO AL422B [37].

1. Wait for a VSYNC event to occur, this will hint that a picture is starting to be taken.
2. Reset the FIFO frame buffer pointer to 0 by sending a pulse to the WRST pin.
3. Set the write enable to active by setting pin WEN high.
4. Wait for another VSYNC event to stop image capturing.
5. Set the write enable pin to low to stop recording data.

The example code is displayed below:



```

1 void CaptureOV7670Frame ()
2 {
3     // 1. Wait for VSync to pulse to indicate the start of the
4     //    image
5     DurationStart = pulseIn(VSYNC, HIGH);
6     // 2. Reset Write Pointer to 0. Which is the beginning of
7     //    frame
8     PulseLowEnabledPin(WRST, 6);
9     // 3. Set FIFO Write Enable to active (high) so that image can
10    //    be written to ram
11    digitalWrite(WEN, HIGH);
12    // 4. Wait for VSync to pulse again to indicate the end of the
13    //    frame capture
14    pulseIn(VSYNC, HIGH);
15    // 5. Set FIFO Write Enable to nonactive (low) so that no more
16    //    images can be written to the ram
17    digitalWrite(WEN, LOW);
18 }

```

Once the picture is saved in the FIFO memory then the **MCU** shall collect it and save it to the SD card, this is accomplished by following this set of instructions [37]:

1. Reset the read FIFO pointer to the start of the frame by setting RRST low and sending 3 pulses to the reading FIFO's clock, RCLK.
2. Send a pulse to the FIFO's reading clock.
3. Read the output data from pins D[7..0]

```

1 void ReadTransmitCapturedFrame ()
2 {
3     // Image file to write to
4     File ImageOutputFile;
5     // Create file in SD
6     ImageOutputFile = SD.open("photo.txt", FILE_WRITE);
7
8     // Set Read Buffer Pointer to start of frame
9     digitalWrite(RRST, LOW);
10    PulsePin(RCLK, 1);
11    PulsePin(RCLK, 1);
12    PulsePin(RCLK, 1);
13    digitalWrite(RRST, HIGH);
14
15    unsigned long ByteCounter = 0;
16    for (int height = 0; height < PHOTO_HEIGHT; height++)
17    {
18        for (int width = 0; width < PHOTO_WIDTH; width++)
19        {
20            for (int bytenumber = 0; bytenumber <
21                PHOTO_BYTES_PER_PIXEL; bytenumber++)

```

```

21     {
22         // Pulse the read clock RCLK to bring in new byte of
           data.
23         // 7ns for RCLK High Pulse Width and Low Pulse Width
           .007 micro secs
24         PulsePin(RCLK, 1);
25         PixelData = PINL;
26         ImageOutputFile.write(PixelData);
27     }
28 }
29 }
30 }
31
32 // Close SD Card File
33 ImageOutputFile.close();
34 }

```

#### 4.2.9 HX-1 Data transmission

The main objective to use this module properly is to generate an [AFSK](#) signal containing the [APRS](#) data. This is achieved by the Timer/Counter unit in the [Arduino](#) MEGA as explained in [C](#).

As it is explained in chapter [B](#), the maximum transfer data in a [AX.25](#) frame is 256B, which is too small for sending images, that is why images won't be sent very often and will very likely just be stored in the SD card.

##### 4.2.9.1 [AX.25](#) frame generation

In order to generate a [AX.25](#) the data headers will be generated as explained in [B](#) and sent, destination and callsign (source) will be:

1. SENDER CALLSIGN = GSA, con SSID = 11
2. DESTINATION CALLSIGN = APRS, con SSID = 0

Once this is set, it will be just needed to set the rest of the headers, that is taken care of by the library provided by trackduino [[17](#)].

##### 4.2.9.2 [APRS](#) frame

In order to send the environmental data collected by the [Arduino APRS](#) will be used. This protocol is really easy since it only specifies that a variables shall be sent as data=value. The following template is selected.

(W/E)00.00(N/S)00.00,MX=00.00,MY=00.00,MZ=00.00,AX=00.00,AY=00.00,AZ=00.00,GX=00.00,0  
where:

- W/E is the West or East latitude angle.
- N/S is the North or South longitude location.
- M- is the [Magnetometer](#) data.
- A- is the [Acceleromter](#) data.
- G- is the [Gyroscope](#) data.
- T is the temperature data.
- P is the pressure data.
- TIME is the time stamp in Hours:Minutes:Seconds.

## CHAPTER

# 5

# VERIFICATION, TESTING AND MANUFACTURING

This chapter details the last step in a product development. During this stage we will check that the final result behaves as it was first detailed in the system requirements [2](#). In order to verify the correct functioning of the system test will be carried testing parts of the system independently. For this purpose we will subdivide this chapter in the electronic/[PCB](#) testing and the software testing.

During this chapter the techniques and prototypes created in order to develop the final product will also be discussed. Therefore this chapter will not follow the orthodox way of presenting an engineering process, since this part should be left only to test the final product while leaving the System Design take care of the prototypes and/or complications that may occur during a product design. I will try to justify this modification by claiming that by doing this a clearer System Design part has been presented by just explaining the in depths of the final product making it easier for the reader to find how a certain system works.

## 5.1 Camera testing

Since the OV7670 module is probably the hardest to install and configure special hardware was created in order to test the connections. The [PCB](#) was created using the [DIY](#) method described in appendix [D](#) The final [PCB](#) follows the schematic connection provided by [\[38\]](#) displayed in figure [5.1](#). The final [PCB](#) made for this purpose is displayed below in figure [5.2](#).

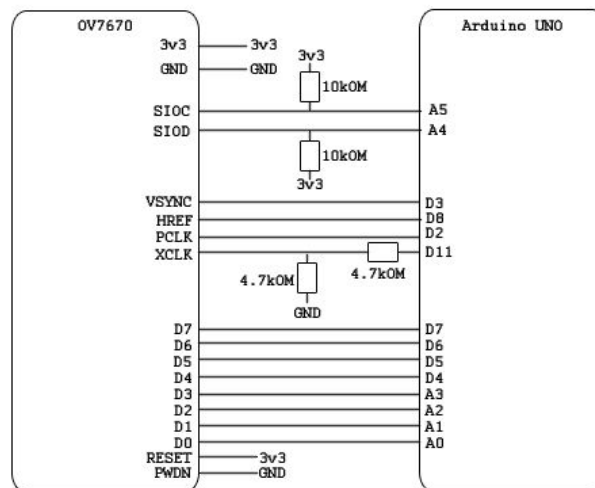


Figure 5.1 – Interconnection between OV7670 and UNO

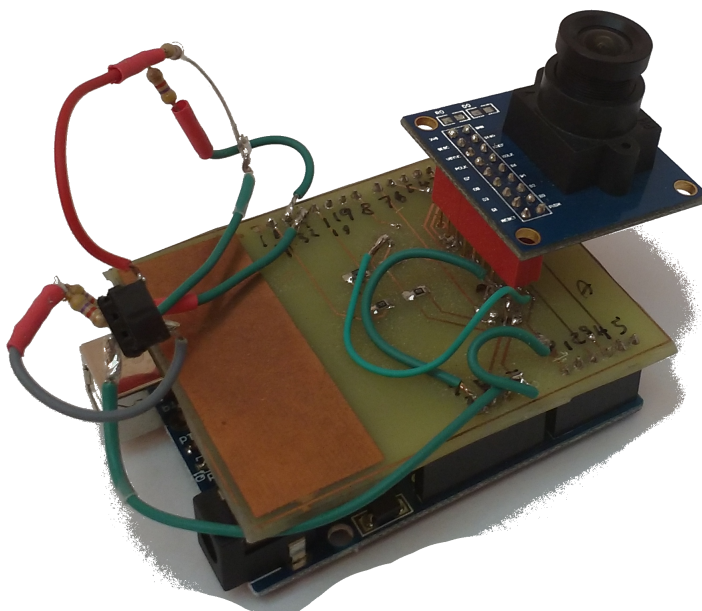


Figure 5.2 – Test board for camera OV7670

After that a low quality photo can be read using a frame grabber from serial to video output, like this one [here](#). The final result is a raw image that looks like the one portrayed in figure 5.3.

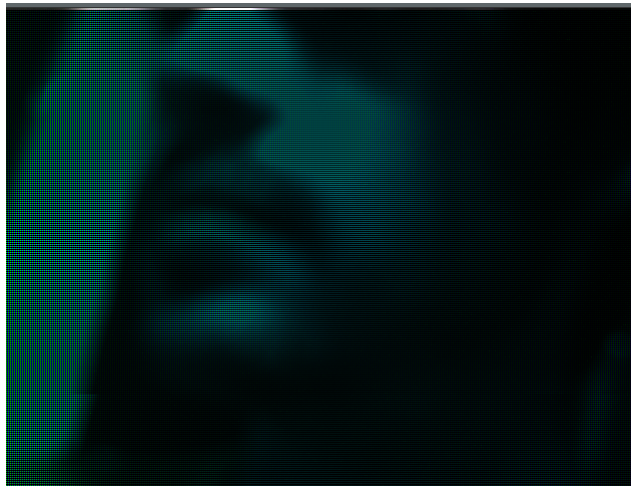


Figure 5.3 – VGA quality image taken with the OV7670

## 5.2 Electronic testing

The main aim of this part is to test that every part of the hardware works and that the interconnection are done in a satisfactory way and that communication within the sensors can be carried away without an issue. Aside from electronic a rule reporting the best way to test if a PCB is well made is to create a prototype. In our case the prototype is presented below, in figure 5.4.

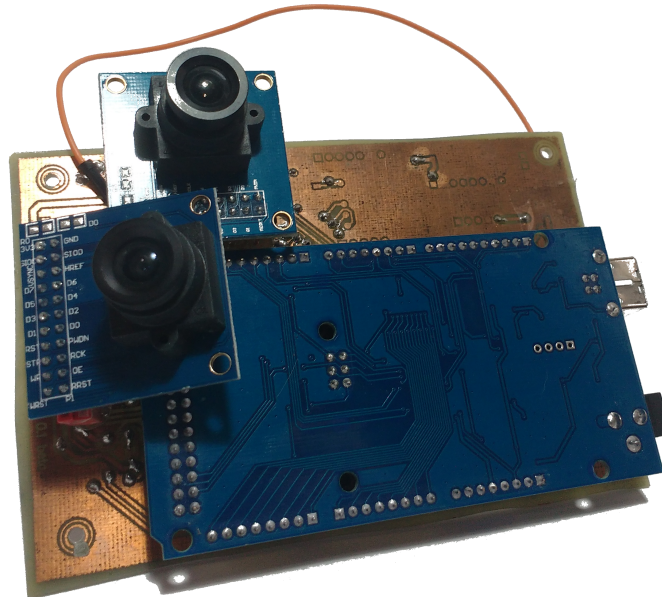
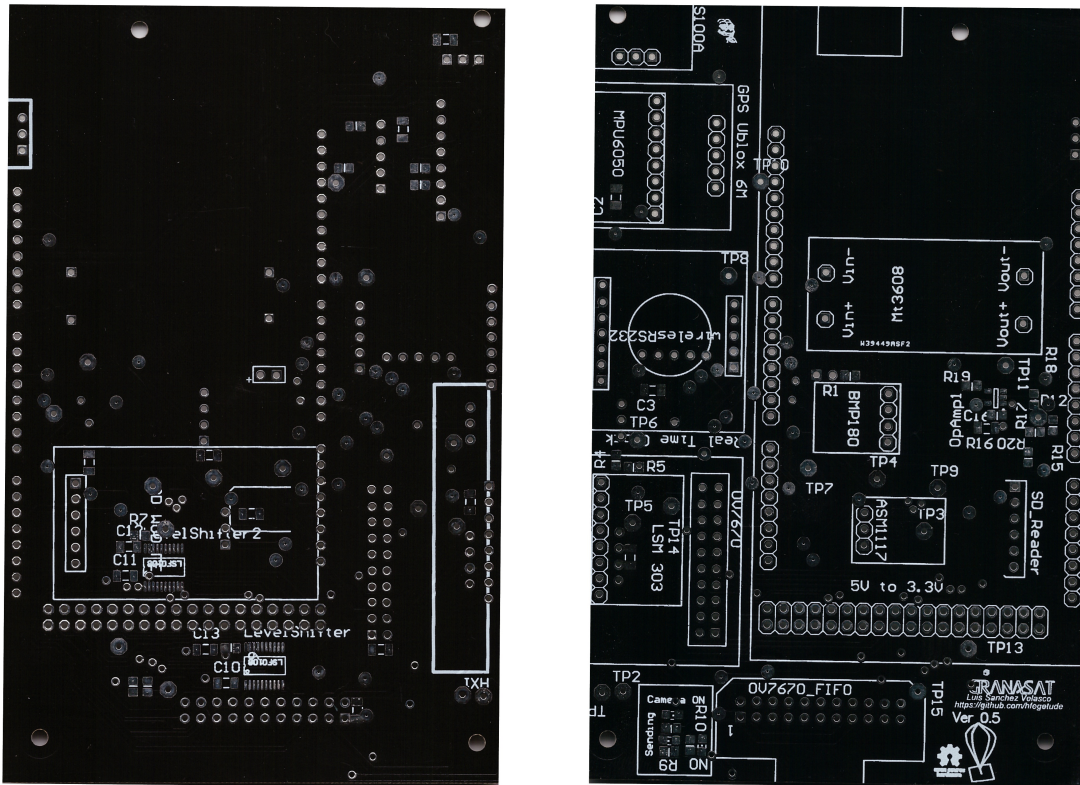


Figure 5.4 – Prototype of the PCB

Once the prototype is revised and connections are satisfactory, the final design can be ordered to an external manufacturer to produce a high quality PCB.



(a) Bottom view of the final PCB

(b) Top view of the final PCB

A full mounted board picture can be seen at figure 5.5

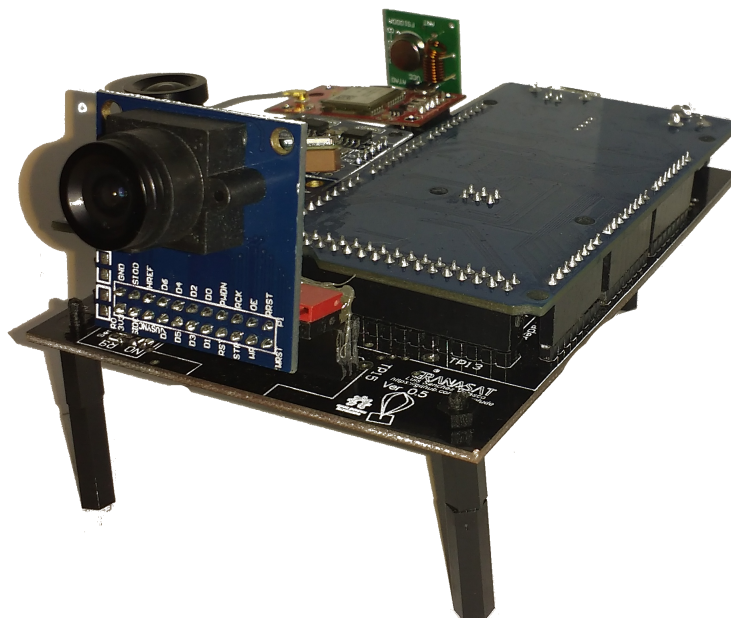


Figure 5.5 – Full minted PCB board

## 5.3 Software testing

We will devoid the testing into different sections, one for each module, in order to make independent tests and face any problems that may occur.

### 5.3.1 BMP180

To test this sensor, temperature measures and pressure measures will be taken and then compared to the results obtained. In order to measure the temperature we will use an infrared thermometer and point it towards the sensor itself, since it is where the measure will be more effective. For the pressure, since no external barometers are available, a weather temperature will be used. The results are show below in figures 5.6c. 5.6b and 5.6a.



(a) Results from the thermometer



(b) Results of the pressure checking, from [foreca.com](#)

```

/dev/tty/USB0
Temp = 33.77 Pressure = 927.01
Temp = 33.78 Pressure = 927.00
Temp = 33.77 Pressure = 926.98
Temp = 33.78 Pressure = 927.04
Temp = 33.78 Pressure = 927.07
Temp = 33.77 Pressure = 926.98
Temp = 33.77 Pressure = 926.99
Temp = 33.79 Pressure = 927.02
Temp = 33.78 Pressure = 927.02
Temp = 33.79 Pressure = 926.99
Temp = 33.78 Pressure = 926.99
Temp = 33.78 Pressure = 926.99
Temp = 33.78 Pressure = 926.98
Temp = 33.78 Pressure = 926.91
Temp = 33.79 Pressure = 926.95
Temp = 33.79 Pressure = 926.99
Temp = 33.79 Pressure = 927.02
Temp = 33.79 Pressure = 927.01
Temp = 33.79 Pressure = 927.00
Temp = 33.79 Pressure = 927.02
Temp = 33.80 Pressure = 927.02
Temp = 33.80 Pressure = 926.99
Temp = 33.79 Pressure = 927.03
Temp = 33.80 Pressure = 927.03
Temp = 33.79 Pressure = 926.99
  
```

(c) Results obtained from the

**Figure 5.6 – BMP180 test results**

As it can be seen the results are quite similar to the ones obtained by external sources, so this test can be considered to be passed.



### 5.3.2 LSM303

In order to test this module we will read the data provided by the device while switching it into different positions, these changes should manifest a change in the resultant vector coordinates, for example, if the current position is with device parallel to the floor, there should be an Z acceleration of about  $9.8 \text{ m} \cdot \text{s}^{-2}$ , if we instead place it perpendicular to the floor, that acceleration should be manifested in the Y axis.

The following data was obtained by exchanging the Z axis whit the X axis. an it's displayed in table 5.1.

Sensor	X	Y	Z	Units
Y + $\pi/2$ rad				
Magnetometer	-0.33	-0.17	0.16	Gauss
Accelerometer	9.89	-0.04	-1.06	$\text{m} \cdot \text{s}^{-2}$
0 rad				
Magnetometer	-0.09	-0.2	-0.43	Gauss
Accelerometer	-0.04	0.51	-10.16	$\text{m} \cdot \text{s}^{-2}$

**Table 5.1** – Data from LSM303 rotating through the Y axis

As it can be seen the leading component rotates along the axis as the board moves, therefore this test is considered passed.

### 5.3.3 MPU6050

Since no proper tool are available to measure the accuracy of the device, we will again try to prove that the device works by leaving it in a rest position and then give the device some rotational along a particular axis by twisting it and then recording the results. These results are displayed in table 5.2.

Sensor	X	Y	Z	Units
Rest				
Gyroscope	0.00	-0.36	-0.04	$^{\circ} \cdot \text{s}^{-1}$
Z rotation				
Gyroscope	0.00	-3.57	57.35	$^{\circ} \cdot \text{s}^{-1}$

**Table 5.2** – Data from MPU-6050 rotating through the Z axis

### 5.3.4 DS1307

In order to test this device we just need to make sure that it can hold the time it passed while the device is unplugged. For that we set the time to 00:00:00 and then we unplug the device. After some time the device will be read again to test if the time measures were effective.

The results were successful, since the clock was set to 00:00:00 and unplugged. After some time has passed it was rechecked to find out that 1 hour has passed.

### 5.3.5 HX-1 Radio Module

In order to test that device a frame will be sent. This test frame will then be read by an SDR and then decoded by an AFSK1200 decoder. SDR# will be used in conjunction with the *rtl-sdr* to be able to read the signal received from the SDR. The resultant signal is then interfaced through a virtual cable to QTMM AFSK decoder.

The result is portrayed in figure 5.7 where a successful read is shown, validating the test.

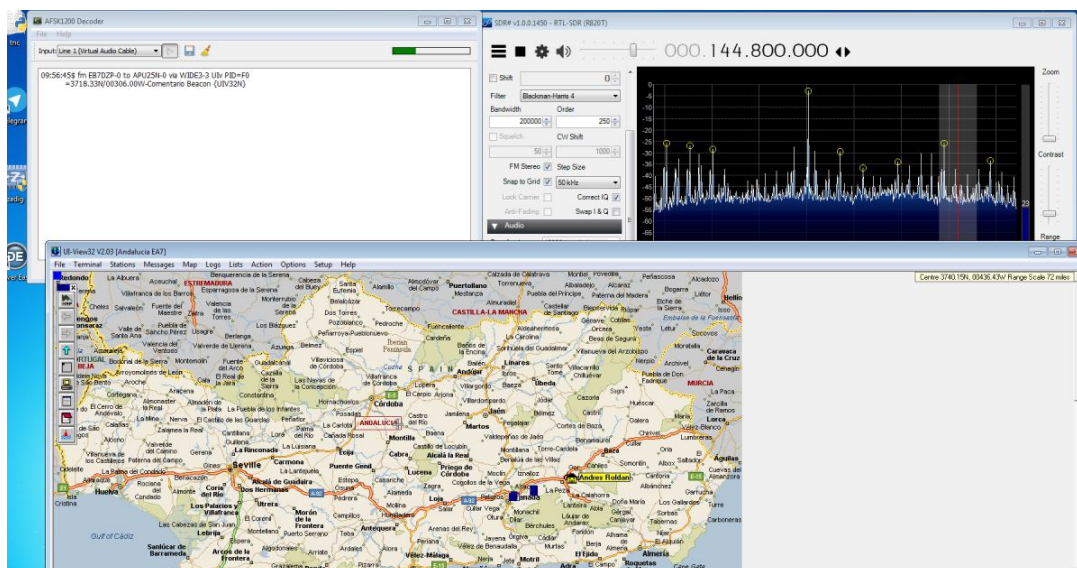


Figure 5.7 – Decoding an AFSK signal



## CHAPTER

# 6

# CONCLUSIONS AND FUTURE LINES

During this document a design it has been presented the design and development of Trackduino, from the first stage to a final product. The focus was made on the software, since it was the main missing part for the project to rise again. It was a challenge to code this device at the same time that the hardware platform was created, inducing many failures due to the lack of understanding of a particular module or the tight time schedules, but in the end to all come to a conclusion.

During this project a lot has been learned about hardware development and manufacturing, with special focus on digital design. From knowing the very basics digital communication protocols such as [UART](#) or [I2C](#) to understand a datasheet from a digital device, and apply this concepts to the board. As long as software goes, many struggles have been defeated, such as coding the register of a [MCU](#) in order to achieve greater functionalities that a library may not offer or making an object-oriented software that makes sense on an small [MCU](#) unit.

Several external modules were included in the [PCB](#). A thermometer and a [Barometer](#) by the name of BMP180, whose connection is not extremely hard but the equations to translate the value obtained into temperature are some kind of tricky to implement. The [Magnetometer](#) and [Acceleromter](#) was also easy to implement, as well as the [Gyroscope](#), the most difficult here, was probably setting the right gain parameter, to obtain sensible numbers to our device's expected performance. The probably most difficult device to implement was

the non FIFO camera, since not much resources are available on-line, and therefore in its implementation a lot of mistakes had to be solved by trial and error and abusive debugging. Time constraints were also an issue since the reading speed needed by the program forced the program to be different from what an average [Arduino C](#) code might look like. Regarding the FIFO camera, its implementation turned out to be a lot easier than expected, due to the amount of very well detailed sources online, as a personal notation on that, it blew my mind how such an easy component as a FIFO can completely transform a product. The [AX.25](#) transmitter was also a challenge, since the code that was written for it was hard to understand and modify to suit the needs from this device, but personally I found very interesting the way a [PWM](#) can achieve an analog transmission, that is the main reason why I decided to make an appendix just for that.

This project also involved a lot of manual soldering since the whole projects, of course, needs to be mounted, that also took a lot of time. Special considerations needed to be taken for a manual [PCB](#) creation. At first, this board was expected to be made in the etching machine available at the university, the LPFK ProtoMat, but since the machine was broken at the last minute, special measures needed to be taken, resulting in the [DIY PCB](#) manufacturing with chemical attack, that ended up being an appendix in the document.

All of these difficulties along with their solutions have made this project possible and entertaining. It can be said that this project was an introduction to the student of how a hardware company works, what is needed and what are the steps to design a successful hardware product. This itself can be considered as a challenge since normally the working routine is very different from the previous one the student may have, forcing her/him to collect information and technical data from a very early stage or produce a lot of work that may be useless. This also provides the student with days where nothing is achieved and days where every is achieved, which is part of what an engineer should be able to face, the fact that not every time the work is going to get done as supposed and unexpected problems may occur.

# 6

## 6.1 Future lines

Along with the creation of this project many paths were open as well to improve the device, but since time is not infinite, at least as we understand it, or not, today, just a certain number of features were selected to be implemented, however these are not incompatible, and will, in fact improve the product, making it more commercial and precise, as well as more easy to use. Some of these features reserved for the future are listed below.

- A Computer receiver program that is able to identify each device and update their status based on [AX.25](#) frames it received.
- Having that computer program, a website could be made, that is able to display the data sent by the device, making it feel like an IoT.
- A communication module with global range, such as GPRS in order to simplify

communication and make it a more portable device.

- The device also incorporates a circuit to measure the battery discharge, giving sensible information about the device life status.
- Two extra radio modules are also included in the [PCB](#), however these are not used, due to time restrictions in the process. These modules offer by one side the ability to communicate with tons of remote objects that use 433.3 MHz ASK modulation, giving the possibility to, maybe, register data on devices as it flight goes. The other module offers the possibility of communicating to the board, making it really interesting to, for example, take pictures on demand.

**6**

## APPENDIX

# A

## WEATHER BALLOON PHYSICS

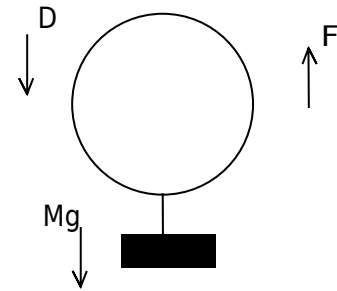
Through this appendix we will be briefly treat the physics of a weather balloon in suspension, so critical parameters such as maximum altitude or flight time until burst occurs can be calculated using some approximations. This appendix is an application of paper [39] to the specifications of the weather balloons sold by [Novalynx](#)<sup>1</sup>. For the drag force occurring at the balloon, equations provided by [40] will be used. The basic balance of forces in the balloon itself can be seen in figure A.1.

Using Newton's second law,  $F = ma$  and assuming no wind, such that the only velocity happens on the  $z$  axis:

$$M \frac{d^2 z}{dt^2} = F - Mg - D \quad (\text{A.0.1})$$

Where:

- $D$  is the drag force, from [40]:  
$$D = \frac{1}{2} c_{D0} \rho_{air} \pi r_0^2 \left( \frac{dz}{dt} \right)$$
- $F$  is the upwards force created by the difference in density. (Buoyant Archimedes Force):  $F = \rho_{air} V_{balloon} g = \rho_{air} \frac{4}{3} \pi r_o^3 = M_{air} g$
- $M$  is the total mass of the balloon.



**Figure A.1** – Balance of forces on the balloon.

<sup>1</sup>This product was mainly chosen to the data provided by the seller, making calculations relatively easier.



## A.1 Maximum payload weight

By removing all dynamic dependent variables in A.0.1 it is possible to calculate the maximum weight as:

$$\rho_{air} \frac{4}{3} \pi r_o^3 g = Mg \quad (\text{A.1.1})$$

## A.2 Weather balloon velocity

Then we will assume that a terminal velocity will be acquired, during most of the ascension at least, the effects of the varying pressure as the altitude increases will be neglected, again, this is an approximation to estimate the total time *on air*.

To obtain the terminal velocity we will set the acceleration  $\frac{d^2z}{dt^2} = 0$  and then by rearranging the equation we can obtain:

$$v_{balloon} = \sqrt{\left(1 - \frac{M}{M_{air}}\right) \frac{8r_o g}{3C_{D0}}} \quad (\text{A.2.1})$$

## A.3 Weather balloon volume

In order to estimate when the balloon will pop we will start by the assumption that it behaves as an ideal gas, that is the famous ideal gas formula:

$$P(z) = \rho_{gas} R_s T(z) \quad (\text{A.3.1})$$

Where we assume the balloon to be a perfect sphere, being:

$$\rho_{gas} = \frac{m_{gas}}{\frac{4}{3} \pi r^3(z)} \quad (\text{A.3.2})$$

It can be noted that the pressure is dependent on the altitude, such relation can be found applying the hydro static equation to the air:

$$\frac{\partial P}{\partial z} = -g \rho_{air} \quad (\text{A.3.3})$$

And then by assuming the atmosphere an ideal gas:

$$P = \rho_{air} R_s T(z) \quad (\text{A.3.4})$$

We see again that for the previous equation that the model states a varying temperature

against the altitude, to accomplish that a simple linear method will be used:

$$T(z) = T'_0 z + T_0 \quad (\text{A.3.5})$$

Then by substituting A.3.5 into A.3.4 then substituting  $\rho_{air}$ , rearranging and integrating on both sides, we can obtain:

$$\int_{P_0}^P \frac{1}{P} dP = \frac{-g}{R_s} \int_0^z \frac{1}{T_0 - T'_0 z} dz \quad (\text{A.3.6})$$

That solves into:

$$P(z) = P_0 \left(1 - \frac{T'_0 z}{T_0}\right) e^{\frac{g}{R_s T'_0} z} \quad (\text{A.3.7})$$

And finally putting A.3.7 into the hydrstatic equation for the balloon A.3.3 and A.3.2 we obtain the relationship between the radius of the sphere and the altitude as:

$$r_{balloon}^3 = \frac{m_{gas} R_{gas}}{\frac{4}{3}\pi P_0} e^{-\frac{g}{R_{air} T'_0} z} \left(\frac{T_0^2 - T_0 T'_0 z}{T_0 - T'_0 z}\right) \quad (\text{A.3.8})$$

## A.4 Applying it to the 400-8210 weather balloon with helium

For this balloon the following data can be obtained:

Nominal weight (gm)	Std Inflated Diameter (cm)	Burst Diameter (cm)
10	46	60

**Table A.1** – Relevant data for 400-8210 weather balloon

Then the solution for z in equation A.3.8 can be obtained with the following data:

- $r_{balloon}$ , that is the burst balloon radius, from table A.1:  $r_{balloon} = 0.3m$ .
- $m_{gas}$ , that is the mass of the gas once the balloon is inflated, that is:  $m_{gas} = \rho_{helium} \frac{2}{3}\pi r^3$ , with a radius of 0.23 m (see table A.1), obtaining: 13.14g
- $R_{helium}$  is  $2077J(kg \cdot K)^{-1}$ .
- $R_{air}$  is  $287J(kg \cdot K)^{-1}$ .
- $g$  is 9.82 m/s.
- $P_0$  is the pressure at sea level 101,320 Pa
- $T_0$  is the global mean temperature, 288K

---

<sup>2</sup>Helium density  $0.1786kg \times m^{-3}$

- $T'_0$  is  $6.45K \cdot km^{-1}$

Where:

$$z = \frac{T_0^2 - T_A T_0}{T'_0(T_0 - T_A)} = 42km; T_A = \frac{\frac{4}{3}\pi r_{balloon}^3 P_0}{m_{gas} R_{gas}} e^{\frac{g}{R_{air} T'_0}} \quad (A.4.1)$$

Now the maximum weight of the payload will be obtained using expression A.1.1, where we can find:

Having a total altitude of 42km, it is possible to calculate the time to reach that high, using expression A.2.1. Where:

- $\rho_{air}$  is assumed to be  $1.1455kg \cdot m^{-3}$
- $r$  is the radius of the inflated balloon, 0.23m
- $M$  is the total mass of the system:  $M = M_{payload} + M_{balloon}$ , in this equation  $M_{balloon} = 10g$

By solving A.2.1 we obtain:

$$M_{payload} = \frac{\rho_{air} \cdot 4 \cdot \pi r_o^3}{3} - M_{balloon} = M_{air} - M_{balloon} = 0.048kg \quad (A.4.2)$$

And finally the total travel time will be the time the balloon will take to reach the maximum height, which was found to be  $h = 42km$  in A.4.1. Using the simple  $t = h/v_{balloon}$ , obtaining  $v_{balloon}$  from A.2.1, identifying the following parameters:

- $C_{D0}$  is the drag coefficient, which for a sphere is found to be 0.1.
- $M_{air}$  is the displaced mass of air,  $\frac{\rho_{air} \cdot 4 \cdot \pi r_o^3}{3} = 0.058kg$
- $M$  will be expected as half of the maximum mass, 0.024kg.

Using A.2.1:

$$v_{balloon} = \sqrt{\left(1 - \frac{1}{2}\right) \frac{8r_0 g}{3c_{D0}}} = 5.48m \cdot s^{-1} \quad (A.4.3)$$

And the total time:

$$t = 42000/5.48 = 7664s \rightarrow 2.12h \quad (A.4.4)$$

## APPENDIX

# B

## BASIC AX.25 DOCUMENTATION

This appendix will go through the basics of the [AX.25](#) protocol. This protocol will try to provide an easy to use link-layer protocol layer between stations that is also capable of broadcasting data.

### B.1 Frame structure

Several structures are possible, depending on the sender needs, them being:

1. Information frame.
2. Supervisory frame.
3. Unnumbered frame.

Though, during this appendix the focus will be placed in the information frame since it will be the one that the device will broadcast. The rest of them are really useful in *Full duplex* communications, which is not the actual case for the device under design.

The frame structure goes as follows in table [B.1](#).

Flag	Address	Control	PID	Info	FCS	Flag
0x7E	112/224 Bits	8/16 Bits	8 Bits	Info N*8 Bits	16 Bits	0x7E

**Table B.1** – *Information frame construction*

### B.1.1 Flag field

In this field an octet flag that delimits the start and the end of the frame is defined such that the receiver can know the beginning and the end of a frame. As it will be seen in the future it could happen that an inside data byte is 0x7E, such byte should be avoided by *bit stuffing*.

#### B.1.1.1 Address field

This field contains the identifier for both the source and the destination.

#### B.1.1.2 Control field

This contains the sequence number of the last frame received, if it existed, and the sequence number of the current sent frame.

#### B.1.1.3 PID field

This will identify the protocol above .

#### B.1.1.4 Information field

This field contains the data that is being sent, it can be no longer than 256 octets, 2048 bits, before bit stuffing is applied.

#### B.1.1.5 Frame Check Sequence

This field will contain a number that assures that the message was not corrupted.

## APPENDIX

### C

# GENERATING AN AFSK SIGNAL WITH ARDUINO

Since the HX-1 module does not include any modulation options, it will just raise the signal's frequency to the desired, a way to produce such signal in base band, also known as AFSK, is needed. The way this signal is generated will be described below briefly, as coded in the Trackduino module [17].

In order to obtain a voltage varying signal with time a [PWM](#) output will be used, this unit is controlled by a counter that is reset when the counter reaches it's peak value. By loading a number into this counter we are able to control the duration of the on state of the output. This generating a voltage that is fast and easy to change on desire.

### C.1 Timer Counter Setup

The counter needs to:

1. Resets once it reaches its peak value.
2. Sets low on compare match.
3. Load a new matching parameter.

The first two specifications can be achieved by configuring the clock with this parameters.

```

1 //Clear on compare match.
2 TCCR3A |= (1 << COM3A1) | (1 << WGM30);
3 //No prescaler on clock source
4 //Fast PWM with 8-bit resolution counter
5 TCCR3B = ( TCCR3B & ~(_BV(CS32) | _BV(CS31) | _BV(WGM33)) ) | (1
  << CS30) | _BV(WGM32);
6 //Resets counter
7 TCNT3 = 0;
8 // Declare pin 5 as output
9 DDRE |= (1 << PORTE3);

```

For loading a new matching value we can write this new value to register OCR3A.

## 3

## C.2 PWM generation

To achieve this part we need to be able to produce two different sine signals, one at 1200 Hz and the other one 2200 Hz, to signalize a zero being transmitted or a one. First we need to define two key constants:

- **Playback Rate:** That is the rate at the device will output a certain voltage level, in our case this will be  $16 \text{ MHz} / 256 = 62500$ , due to the counter have to reach up to 256 before loading a new sample.
- **Baud rate:** The symbols per second we want our device to be able to send, since its [AFSK1200](#), this value will be 1200.

With this two values we can obtain the number of samples a single byte should take as:

$$\text{Samples per Baud} = \frac{\text{Playback Rate}}{\text{Baud Rate}} \quad (\text{C.2.1})$$

In order to calculate the instant phase of the signal so that it can be outputted, for that purpose we need the phase delta, that is, the change in the phase in each sample outputted. That can be computed as:

$$\Delta\delta = \frac{f}{\text{Playback Rate}} \quad (\text{C.2.2})$$

And the phase for each sample can be computed as:

$$\delta \leftarrow \delta + \Delta\delta \quad (\text{C.2.3})$$

The last thing we need is a table that given a phase value it can return the timer value.

For example for a 9bit table (512 values), for any given phase, the table value will be:

$$x = 512 \cos \delta \quad (\text{C.2.4})$$

The main algorithm for loading a buffer with samples, given the last explained values and a new byte is described below.

1. Load a new bite from data.
2. Check if it's one or zero to change phase delta to phase delta 1200 or phase delta 2200.
3. Compute phase as  $\text{phase} = \text{phase delta} + \text{phase}$ .
4. Obtain the sample value by checking the table for the phase value.
5. Load value into sample buffer.
6. Check if enough samples have been generated for this bit, if so grab a new bit, otherwise return to step 3.



**3**

## APPENDIX

# D

## DIY PCB MANUFACTURING

During this appendix the steps followed to manufacture the [PCB](#) and the items used to achieve this goal will be described and briefly explained as a tutorial.

### D.1 Transferring the stencil to the copper

In order to create a satisfactory [PCB](#) the image holding the traces shall be printed on to the cooper, to achieve this a special paper is used. For the prototypes the [PulsarPro: PCB Toner Transfer Paper](#) was used, a picture is show in fig. [D.1](#).

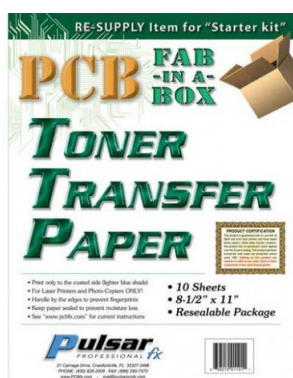


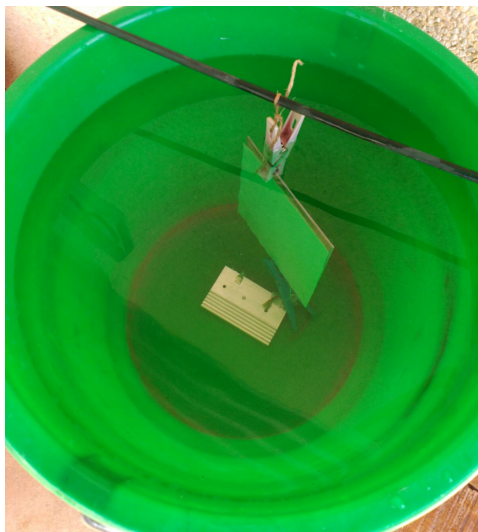
Figure D.1 – Paper used to print the [PCB](#)

Once the paper is printed, the next step is to place the paper carefully onto the PCB and make sure that both sides are placed exactly aligned in order for the pins to coincide. For that is really useful to set a point of reference points in the PCB and drill them, so for example, a nail can be used to math both top and bottom. After that, heat should be added to the board with the stencil paper, for that a plastifier whose thermostat was removed (in order to apply more heat) was used. This process is seeing in figure



**Figure D.2** – *Transferring the stencil to the PCB*

When enough heat was applied it's time to remove the paper, in order to do that the PCB is submerged into water (figure D.3). This process will automatically remove the stencil, so no external forces are needed, resulting in a more satisfactory transaction.



**Figure D.3** – *Submerging the PCB into water*

The resultant **PCB** with the stencil printed on it will look something like the one displayed in figure D.4.

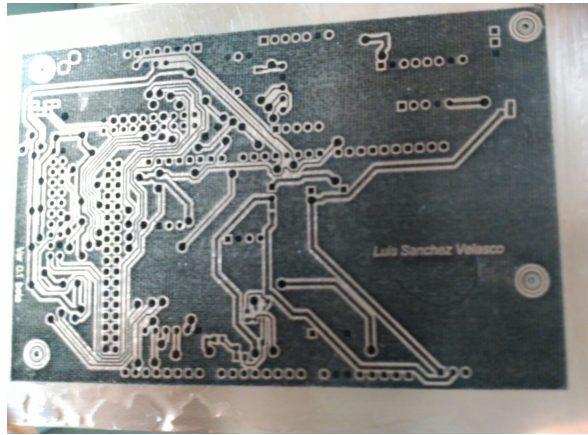


Figure D.4 – Resultant **PCB** from the stencil transfer process

## D.2 Removing the surplus copper

To achieve this we will use the chemical reaction D.2.1.



To achieve this mixture will be created with the same proportion of a hydrochloric acid, hydrogen peroxide and water. The board will then be submerged in the resultant solution, as portrayed in figure D.5

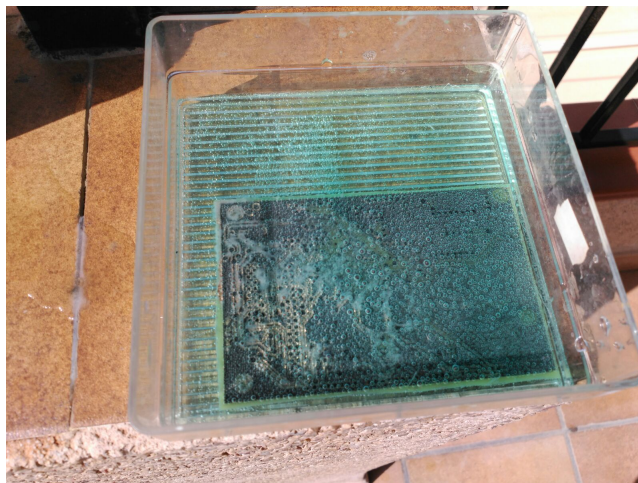
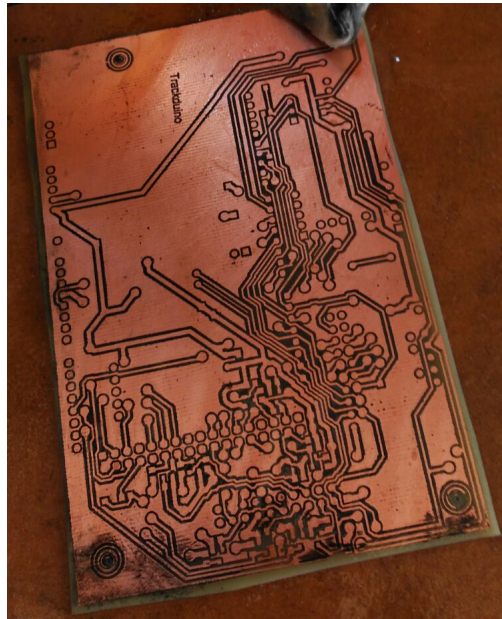


Figure D.5 – Submerging the board in the copper solvent solution

After that, the leftover ink can be removed with alcohol, obtaining a **PCB** as displayed in figure **D.6**, with just the drills to be fully finished.



**Figure D.6** – *Final prototype PCB*

## APPENDIX

# E

## PROJECT BUDGET

### E.1 Electronic Cost

This section describe the cost of producing and manufacturing the [PCB](#), taking in count both the final [PCB](#) and the prototypes made. Table [E.1](#) sums up that costs. Electronic

Item	Cost €
Copper plate	5
Chemicals	3
Manufactured <a href="#">PCB</a>	37
<b>TOTAL</b>	45

**Table E.1** – *Electronic cost of the [PCB](#)*

components are considered in table [E.4](#).

### E.2 Software

This section will describe the prices of the software licenses used, listed in table [E.2](#).

Software	Owner	Cost €
Altium designer 14.3	UGR	0
Atom	Open Source	0
Arduino IDE	Open Source	0
VIM	Open Source	$-\infty$ <sup>42</sup>
Inkscape	Open Source	0
TexLive	Open Source	0
<b>TOTAL</b>		0

**Table E.2** – *Software cost of product*

### E.3 Human Resources

Since this work is part of a Bachelor Thesis, we will consider the case where a junior engineer was contracted for this project, working 9 months full time, for about 10 €/ hour. Another senior engineer was contracted as a supervisor for 50 €/ hour. Table represents the full cost. This is presented in table E.3.

Charge	Hours	Cost €
Junior Engineer	1440	14400
Senior Engineer	180	9000
<b>TOTAL</b>		23400

**Table E.3** – *Human cost of product*

Item	Description	Units	Cost/ Units(€)	Used Cost (€)
Arduino MEGA	Microcontroller board	1	5.1	2.55
MT3608	DC-DC converter	1	7.66	7.66
HX.1	Radio Module	1	32	32
SD adapter		1	3	3
SD card	4 Gigabytes	1	4	4
MPU6050	Gyroscope + Accelerometer	1	2.67	2.67
LSM303DLHC	Accelerometer + Magnetometer	1	4.76	4.76
BMP180	Barometer + Thermometer	1	1.3	1.3
U-BLOC GPS	GPS module	1	6.76	6.76
OV7670	Camera	1	4.76	4.76
OV7670 FIFO	Camera	1	9.76	9.76
tinyRTC	Real Time Clock	1	4	4
FS1000A	Radio Module	1	1.6	1.6
Battery	Lithium battery	1	6	6
LSF0108	Voltage level translator	2	0.7	1.4
SMD0805 Resistor	Various values	14	0.02	0.28
SMD0805 Capacitor	Various values	12	0.02	0.24
Leds	Various Colors	3	0.2	0.6
			<b>Total</b>	<b>93.34</b>

Table E.4 – Component cost





# REFERENCES

- [1] Duracell, "Ultra power 9v." Datasheet. [http://www.celltech.fi/fileadmin/user\\_upload/Celltech/Prod.sheets/Duracell\\_Ultra-Power\\_9V.pdf](http://www.celltech.fi/fileadmin/user_upload/Celltech/Prod.sheets/Duracell_Ultra-Power_9V.pdf).
- [2] Energizer, "E92." Datasheet. <http://data.energizer.com/pdfs/e92.pdf>.
- [3] Adafruit, "Usb liion/lipoly charger - v1.2," 2005.
- [4] "Silicon bandgap temperature sensor." Wikipedia. [https://en.wikipedia.org/wiki/Silicon\\_bandgap\\_temperature\\_sensor](https://en.wikipedia.org/wiki/Silicon_bandgap_temperature_sensor).
- [5] W. Commons, "Piezoelectric sensor frequency response," 2007. File: LambdaPlaques.jpg.
- [6] A. Chulliat, S. Macmillan, P. Alken, C. Beggan, M. Nair, B. Hamilton, A. Woods, V. Ridley, S. Maus, and A. Thomson, "The us/uk world magnetic model for 2015-2020," *NOAA*, 2015.
- [7] Atmel, *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V*, revision b ed., May 2015. [http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561\\_datasheet.pdf](http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf).
- [8] "Ov7670 schematic." WebPage. <http://www.lxxtech.com/alientek-ov7670-camera-module-with-fifo-stm32f103-development-board-driver-p818.html>.
- [9] STMicroelectronics, *LSM303DLHC:Ultra-compact high-performance eCompass module:3D accelerometer and 3D magnetometer*, November 2013. Available

## References

- at <http://www.st.com/web/en/resource/technical/document/datasheet/DM00027543.pdf>.
- [10] InvenSense Inc., 1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A. Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104, *MPU-6000 and MPU-6050 Product Specification*, revision 3.3 ed., May 2012. .
- [11] u blox, "Neo-6." Datasheet. [https://www.u-blox.com/sites/default/files/products/documents/NEO-6\\_DataSheet\\_\(GPS.G6-HW-09005\).pdf](https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf).
- [12] OmniVision, "Ov7670." Datasheet. <https://www.voti.nl/docs/OV7670.pdf>.
- [13] Radiometrix, "Hx1." Datasheet. <http://www.radiometrix.com/files/additional/hx1.pdf>.
- [14] Mantech, "fs1000a." Datasheet. [http://www.mantech.co.za/Datasheets/Products/433Mhz\\_RF-TX&RX.pdf](http://www.mantech.co.za/Datasheets/Products/433Mhz_RF-TX&RX.pdf).
- [15] Elecrow, "Hc-12." Datasheet. <https://www.elecrow.com/download/HC-12.pdf>.
- [16] F. J. L. Lorente, "Design, development and testing for low cost systems for satellite image reception," bsc thesis (dissertation), University of Granada, Granada, May 2016. This BSc Thesis is a contribution to the GranaSAT project.
- [17] "Trackuino official webpage." Web page. <http://www.trackuino.org/>.
- [18] "University of Louisiana weather balloon projects." Web page. <http://ee.louisiana.edu/research/cape/weather-balloon-projects>.
- [19] TAPR, *AX.25 Link Access Protocol for Amateur Packet Radio*, July 1998. <https://www.tapr.org/pdf/AX25.2.2.pdf>.
- [20] D. Deng, "Lion batteries: basics, progress, and challenges," *Energy Science & Engineering*, 2012.
- [21] L. Ada, "Li-ion & lipoly batteries," *Adafruit Webpage*, 2016.
- [22] W. Commons, "Li ion battery from a laptop computer.," 2010. File: Li ion laptop battery.jpg.
- [23] W. Commons, "Bateria comercial de li-ion," 2005. File: Lithium Battery1.jpg.
- [24] F. Semiconductor, "Xtrinsic mpl3115a2 i2c precision altimeter." Datasheet. [https://cdn-shop.adafruit.com/datasheets/1893\\_datasheet.pdf](https://cdn-shop.adafruit.com/datasheets/1893_datasheet.pdf).

- [25] BOSCH, "Bmp280." Datasheet. [https://ae-bst.resource.bosch.com/media/\\_tech/media/datasheets/BST-BMP280-DS001-18.pdf](https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMP280-DS001-18.pdf).
- [26] BOSCH, "Bmp180." Datasheet. <https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>.
- [27] J. Lenz and A. S. Edelstein, "Magnetic sensors and their applications," *IEEE SENSORS JOURNAL*, 2006.
- [28] D. DiLella, "A micromachined magnetic-field sensor based on an electron tunneling displacement transducer," *Sensors and Actuators*, 2000.
- [29] S. studio, "fs1000a." Datasheet. [http://www.mouser.com/catalog/specsheets/Seeed\\_113060000.pdf](http://www.mouser.com/catalog/specsheets/Seeed_113060000.pdf).
- [30] Edgi, "Gy-511 - 3-axis accelerometer and 3-axis magnetometer [lsm303dlhc]." Edgi Foundation Webpage. [http://edgi.ru/moduli\\_datchiki\\_arduino/](http://edgi.ru/moduli_datchiki_arduino/).
- [31] M. Integrated, "Ds1307." Datasheet. <https://datasheets.maximintegrated.com/en/ds/DS1307.pdf>.
- [32] Olimex, "Mt3608." Datasheet. <https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf>.
- [33] Texas Instruments, *Voltage-Level Translation With the LSF Family*, revision b ed., May 2015. <http://www.ti.com/lit/an/slva675b/slva675b.pdf>.
- [34] Open source, *Bosch BMP085 Barometer Floating Point Pressure Calculations*, 1 ed., January 2013. <http://wmx00.sourceforge.net/Arduino/BMP085-Calcs.pdf>.
- [35] "Rmc sentence information." Web page. <http://www.gpsinformation.org/dale/nmea.htm#RMC>.
- [36] "The linux ov7670 driver." WebPage. <http://elixir.free-electrons.com/linux/latest/source/drivers/media/i2c/ov7670.c>.
- [37] Roberet Chin, *Beginning Arduino OV7670 camera development*, revision a ed., 2015.
- [38] "Ov7670 arduino uno connection." WebPage. <http://privateblog.info/arduino-uno-i-kamera-ov7670-primer-ispolzovaniya/>.
- [39] M. Denny, "Weather balloon ascent rate," *Phys. Teach.*, 2016.
- [40] "Drag of a sphere." Web page. <https://www.grc.nasa.gov/www/k-12/airplane/dragSphere.html>.