



The aim of this project is to improve and program the Testbed for its use with the CubeSat GranaSAT. Specifically, the Testbed will be programmed to use it as a light detector. These kind of Testbeds, based on an air-bearing, have an expensive cost to be acquired for the GranaSAT project, so we have the intention to develop a low-cost Testbed for a Cubesat 1U platform. Moreover, in this project we will study and analyze the SDR systems used by the GranaSAT team to receive signals and we will measure the SDR's accuracy.



Antonio José Ortiz Sánchez is a telecommunication engineer from Fuente del Maestre, Badajoz, Spain. He finished his Master's studies in 2016 at University of Granada. He starts working with GranaSAT Team in 2015, where he did his final master project continuing the developing of the Testbed for the GranaSAT Cubesat and specialising his studies in the aerospace field.



Andrés María Roldán Aranda is the academic head of the GranaSAT development team, and the tutor of this project. He is professor in Department of Electronics and Computer Technology at University of Granada.

FINAL MASTER  
PROJECT

Antonio José Ortiz Sánchez An improved Testbed for a 1U Cubesat

TELECOMMUNICATIONS  
ENGINEERING

2015/16



# UNIVERSITY OF GRANADA

## Master in Telecommunications Engineering



FINAL MASTER PROJECT

## An improved Testbed for a 1U Cubesat

Antonio José Ortiz Sánchez  
Academic year 2015/2016

Tutor: Andrés María Roldán Aranda



ugr

Universidad  
de Granada

MÁSTER EN INGENIERÍA DE TELECOMUNICACIONES  
TRABAJO FIN DE MÁSTER

*“An improved Testbed for a  
1U CubeSAT”*

CURSO: 2015/2016

ANTONIO JOSÉ ORTIZ SÁNCHEZ

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—  
Granada, 16 de Septiembre de 2016





ugr

Universidad  
de Granada

MÁSTER EN INGENIERÍA DE TELECOMUNICACIONES  
TRABAJO FIN DE MÁSTER

*“An improved Testbed for a  
1U CubeSAT”*

REALIZADO POR:

Antonio José Ortiz Sánchez

DIRIGIDO POR:

Andrés Maria Roldán Aranda

DEPARTAMENTO:

Electrónica y Tecnología de los Computadores



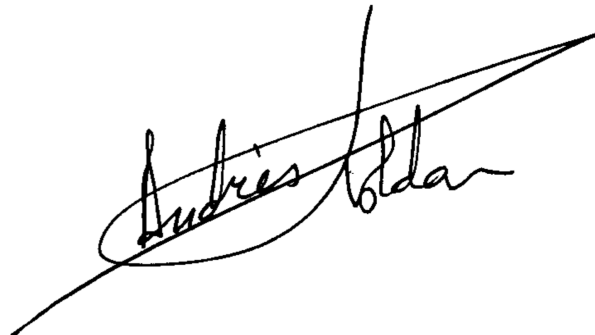
D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Máster de D. Antonio José Ortiz Sánchez,

Informa que el presente trabajo, titulado:

***“An improved Testbed for a 1U CubeSAT”***

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

Granada a 16 de septiembre de 2016

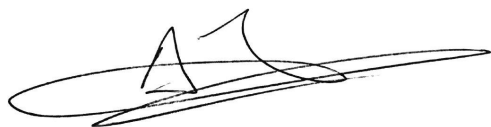
A handwritten signature in black ink, reading "Andrés Roldán", with a long horizontal stroke extending to the right.

Fdo. Andrés María Roldán Aranda




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

Granada a 18 de Diciembre de 2014

A handwritten signature in black ink, appearing to be 'A. J. O. S.', written over a horizontal line.

Fdo. Antonio José Ortiz Sánchez

A handwritten signature in black ink, appearing to be 'Andrés Roldán', written over a horizontal line.

Fdo. Andrés María Roldán Aranda





# **An improved Testbed for a 1U CubeSAT**

**Antonio José Ortiz Sánchez**

## **PALABRAS CLAVE:**

Testbed, cubesat, picosatélite, GranaSAT, arduino, acelerómetro, magnetómetro, rodamiento, aire, sin fricción, lowcost, MATLAB, altium, inercia, ruedas, giróscopo, análisis, diseño, electrónica, PCB, ADCS, detector de luz, LDR, SDR, FUNcube, SDR#, RTL, precision, SINAD, analizador de comunicaciones.

## **RESUMEN:**

Los Cubesats dan a los estudiantes la oportunidad de tener una experiencia práctica diseñando, implementando y trabajando con la operación de un proyecto aeroespacial real. El objetivo de este proyecto es continuar el trabajo de anteriores miembros de GranaSAT en el diseño y construcción de un Testbed para su uso con el CubeSat GranaSAT, un picosatélite diseñado e implementado por alumnos de la Universidad de Granada. Este tipo de Testbeds, basados en un rodamiento de aire, tienen un coste muy elevado para ser adquirido por el proyecto GranaSAT, por ello, tenemos la intención de fabricar un Testbed de bajo coste para una plataforma Cubesat 1U.

En concreto se van a solucionar algunos aspectos del Testbed como el problema de la fuente de alimentación y se programará para utilizarlo como detector de luz, de manera que el Testbed se oriente hacia una fuente de luz.

Por otro lado, en este proyecto también se estudiará y analizarán los sistemas SDR (RTL-SDR and FUNcube Dongle) que los miembros de GranaSAT utilizan para la recepción de señales y se medirá su precisión.



**KEYWORDS:**

Testbed, cubesat, GranaSAT, arduino, accelerometer, magnetometer, airbearing, bearing, frictionless, lowcost, MATLAB, altium, reaction, wheels, reactionwheels, gyroscope, analysis, design, electronics, PCB, ADCS, LDR, light detector, FUNcube, SDR#, RTL, accuracy, SINAD, communication test set.

**ABSTRACT:**

Cubesats allow students the possibility to have a true hands-on experience with the design, implementation and operation of a real aerospace experience. The aim of this project is to continue the work made by others GranaSAT members in the design and building of a Testbed for its use within the CubeSat GranaSAT, picosatellite designed and implemented by students of the University of Granada. That kind of Testbeds, based on an air-bearing, have a expensive cost to be acquired for the GranaSAT project, so we have the intention to develop a low-cost Testbed for a Cubesat 1U platform.

Specifically, some problems of the Testbed are going to be solved as the problem with the power supply and the Testbed will be programmed as a light detector, so that the Testbed will be oriented face to the light source.

On the other and, in this project we will study and analyze the SDR systems (RTL-SDR and FUNcube Dongle) that are being used by the GranaSAT memberships for the signal reception and its accuracy will be measured.



*Dedicado*

*A mis abuelos Julia y Manolo, que  
estaban deseando que finalizara mis estudios*



## *Agradecimientos:*

En primer lugar, a mis padres Lucía y Jose Antonio, a mi hermana Lucía, a Mari Carmen y a todos los demás miembros de mi familia, abuelos, tíos y primos, me gustaría agradecerles su apoyo incondicional, tanto en mi etapa de estudiante como en la vida en general.

También quiero dedicar unas palabras de agradecimiento a todos los amigos y compañeros que me han acompañado a lo largo de todos estos años como estudiante y que han conseguido que se convierta en una etapa inolvidable. Mención especial para mis compañeros de GranaSAT. Trabajar con ellos ha hecho que disfrute realizando este proyecto.

Por ultimo, agradecimiento especial a Andrés Roldán, por su tiempo dedicado y por sus consejos para este proyecto y para mi futura vida profesional.

A todos ellos, gracias de corazón.

## *Acknowledgments:*

Firstly, to my parents Lucia and Jose Antonio, to my sister Lucía, to Mari Carmen and to the other members of my family grandparents, cousins, uncles and aunts, I would like to thank them for their unconditional support, both in my time as a student and in life in general.

I also want to say a few words of thanks to all the friends and partners who have accompanied me over the years as a student and who have made an unforgettable stage. Special mention to my GranaSAT partners. Working with them has made you enjoy doing this project.

Finally, special thanks to Andrés Roldán for his time spent and advice for this project and for my future professional life.

To all of them, heartfelt thanks.





# INDEX

Autorización Lectura	V
Autorización Depósito Biblioteca	VII
Resumen	IX
Dedicatoria	XIII
Acknowledgments	XV
Index	XVII
List of Figures	XXIII
List of Videos	XXIX
List of Tables	XXXI
Glossary	XXXV

<b>Acronyms</b>	<b>XXXIX</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Objectives and motivation . . . . .	1
1.2 Cubesat background . . . . .	2
1.2.1 Attitude Determination and Control System . . . . .	4
1.2.2 Testbed . . . . .	6
<b>2 Client requests and requirements</b>	<b>7</b>
2.1 Client requests . . . . .	7
2.2 Requirements . . . . .	8
2.2.1 Requirements for performance analysis of the SDR . . . . .	8
2.2.2 Requirements for programming the ADCS . . . . .	8
2.3 Project structure . . . . .	9
<b>3 SDR analysis</b>	<b>13</b>
3.1 RTL-SDR . . . . .	13
3.2 FUNcube Dongle . . . . .	15
3.3 SDR Software. SDR# . . . . .	16
3.4 SDR parameters . . . . .	17
3.5 Accuracy measurement method . . . . .	20
<b>4 Testbed analysis</b>	<b>23</b>
4.1 Air-bearing and air compressor analysis . . . . .	23
4.1.1 Air-bearing analysis . . . . .	23
4.1.2 Air compressor analysis . . . . .	26
4.2 PCB analysis . . . . .	30
4.2.1 Microprocessor analysis . . . . .	32
4.2.2 Display LCD . . . . .	34
4.2.3 Power system analysis . . . . .	34

4.2.4	Sensors analysis . . . . .	35
4.2.4.1	MPU6050 . . . . .	36
4.2.4.1.1	MPU6050 calibration . . . . .	41
4.2.4.2	LSM303 . . . . .	41
4.2.4.3	Light sensors . . . . .	44
4.2.5	Actuators analysis . . . . .	45
4.2.5.1	X axis motor . . . . .	48
4.2.5.2	Y axis motor . . . . .	51
4.2.5.3	Z axis motor . . . . .	54
4.2.6	Wireless communication analysis . . . . .	57
4.3	Additional elements . . . . .	58
4.4	PID controller . . . . .	59
4.4.1	PID tuning . . . . .	60
<b>5</b>	<b>SDR's accuracy measurement</b>	<b>63</b>
5.1	Previous steps . . . . .	63
5.1.1	PPM Measurement and offset correction . . . . .	63
5.1.2	SINAD maximization . . . . .	65
5.2	Accuracy measurement . . . . .	68
5.2.1	Results . . . . .	69
5.2.1.1	RTL-SDR accuracy . . . . .	69
5.2.1.2	FUNcube accuracy . . . . .	72
<b>6</b>	<b>Power supply design and implementation</b>	<b>75</b>
6.1	Battery . . . . .	75
6.2	Step Up . . . . .	76
6.2.1	Step Up design . . . . .	76
6.2.1.1	MAX1674 . . . . .	78
6.2.1.2	MAX8211 . . . . .	79

---

6.2.2	2D view of the PCB design . . . . .	81
6.2.3	3D view of the PCB design . . . . .	84
6.2.4	Step Up final implementation . . . . .	88
6.3	Charger Module . . . . .	89
6.4	Voltage Regulator . . . . .	91
6.5	Final implementation . . . . .	92
<b>7</b>	<b>Software desing and implementation</b>	<b>95</b>
7.1	Software general description . . . . .	95
7.1.1	Watchdog . . . . .	99
7.1.2	Real Time Operating System . . . . .	99
7.2	Initial setup . . . . .	101
7.3	LCD screen . . . . .	102
7.4	Reading Sensors . . . . .	104
7.5	Games . . . . .	105
7.5.1	Light Game . . . . .	105
7.5.2	Shake Game . . . . .	108
7.6	PID controller implementation . . . . .	108
7.6.1	Step response . . . . .	109
7.6.2	Parameter calculation . . . . .	113
7.6.2.1	Rise time . . . . .	113
7.6.2.2	Response time . . . . .	114
7.6.2.3	Final parameter calculation . . . . .	117
7.6.3	Choosing a sampling interval . . . . .	119
7.7	Communication software . . . . .	121
7.8	Interrupt . . . . .	122
<b>8</b>	<b>Results and conclusions</b>	<b>123</b>
8.1	Results . . . . .	123

---

8.2	Future work . . . . .	125
8.3	Conclusions . . . . .	125
	<b>References</b>	<b>127</b>
	<b>A Bootloader installation</b>	<b>133</b>
	<b>B Kalibrate-RTL software</b>	<b>137</b>
	<b>C Project Budget</b>	<b>141</b>
C.1	Electronics costs . . . . .	141
C.2	Software . . . . .	142
C.3	Human Resources . . . . .	143



# LIST OF FIGURES

1.1	Logo GranaSAT . . . . .	2
1.2	Ncube-2, a Norwegian Cubesat [52] . . . . .	2
1.3	Different dimensions for Cubesats [22] . . . . .	3
1.4	Cubesat block diagram [25] . . . . .	3
1.5	ADCS for a satellite [21] . . . . .	4
1.6	Cubesat Delfi-n3Xt actuators . . . . .	5
1.7	ADCS GranaSAT block diagram [21] . . . . .	5
1.8	Testbed designed and built by Victor Burgos . . . . .	6
2.1	Project structure . . . . .	10
3.1	RTL-SDR used in GranaSAT project . . . . .	14
3.2	RTL-SDR block diagram . . . . .	14
3.3	FUNcube Dongle used in GranaSAT project . . . . .	15
3.4	Main view of SDR# . . . . .	17
3.5	RTL-SDR tuning without ppm correction. . . . .	18



3.6	FUNcube Dongle tuning with ppm correction. . . . .	19
3.7	Marconi 2022 signal generator . . . . .	20
3.8	HP 8903B audio analyzer . . . . .	20
3.9	Marconi 2955 communication test set . . . . .	21
3.10	Sensibility measurement schematics . . . . .	21
4.1	Pink air-bearing . . . . .	24
4.2	Green air-bearing . . . . .	24
4.3	Air-bearing repair . . . . .	25
4.4	Air-bearing before and after the reparation . . . . .	25
4.5	Air-bearing make by epoxy resin . . . . .	26
4.6	CP2525 air compressor[20] . . . . .	27
4.7	Graph of the variation of the friction-less scenary time with the weight and the air pressure . . . . .	28
4.8	New compressor used for the air-bearing . . . . .	29
4.9	ATmega328p pin-out [19] . . . . .	32
4.10	Pin-out for ATmega328p and Arduinio boards [1] . . . . .	33
4.11	LCD display and I <sup>2</sup> C converter (red board)[21] . . . . .	34
4.12	MPU-6050 [45] . . . . .	36
4.13	Yaw, pitch and roll reference system [28] . . . . .	39
4.14	Axes used as system reference [28] . . . . .	40
4.15	PCB reference system [21] . . . . .	40
4.16	LSM303 [29] . . . . .	42
4.17	LSM303 reference system [49] . . . . .	42
4.18	Light dependent resistor [33] . . . . .	45
4.19	Photo tachometer [43] . . . . .	46
4.20	Revolutions per minute with PWM pulse in X axis motor . . . . .	50
4.21	Revolutions per minute with PWM pulse in Y axis motor . . . . .	53
4.22	Revolutions per minute with PWM pulse in Z axis motor . . . . .	56

4.23 Xbee device from Digi [26] . . . . .	57
4.24 Buzzer installed in the Testbed . . . . .	58
4.25 Pull up resistor button [21] . . . . .	58
4.26 PID block diagram [53] . . . . .	59
4.27 Step response [35] . . . . .	60
5.1 RTL-SDR tuning with ppm correction. . . . .	64
5.2 Schematic of the conexions. . . . .	65
5.3 Marconi 2955a. . . . .	65
5.4 Screen of the Marconi 2955a without filtering audio. . . . .	66
5.5 Received signal inSDR#. . . . .	67
5.6 Screen of the Marconi 2955a with audio filtering. . . . .	67
5.7 Received signal inSDR# with audio filtering. . . . .	68
5.8 Accuracy of the RTL-SDR device. . . . .	70
5.9 Accuracy average of the RTL-SDR device. . . . .	71
5.10 Accuracy of the FUNcube Dongle device. . . . .	72
5.11 Accuracy average of the FUNcube Dongle device. . . . .	73
5.12 Comparison between RTL-SDR and FUNcube Dongle accuracy. . . . .	74
6.1 Battery MGL2803 . . . . .	76
6.2 MAX1674 device . . . . .	78
6.3 MAX1674 PinOut . . . . .	78
6.4 MAX1674 typical operating circuit for 5 V output . . . . .	79
6.5 MAX8211 device . . . . .	80
6.6 MAX8211 PinOut . . . . .	80
6.7 MAX8211 typical operating circuit . . . . .	81
6.8 3D view of the top side . . . . .	84
6.9 3D view of the bottom side . . . . .	85
6.10 3D view of the top left side . . . . .	85

6.11	3D view of the top right side . . . . .	86
6.12	3D view of the bottom left side . . . . .	86
6.13	3D view of the bottom right side . . . . .	87
6.14	MT3608 2A Max DC-DC Step Up . . . . .	89
6.15	MT3608 typical operating circuit [15] . . . . .	89
6.16	TP4056 Li-Ion battery charger . . . . .	90
6.17	Battery charger behaviour . . . . .	90
6.18	LM1117 Regulator [18] . . . . .	91
6.19	LM1117 PinOut [37] . . . . .	92
6.20	Female and male headers . . . . .	92
6.21	Switches . . . . .	93
6.22	System power before adding it to the PCB . . . . .	93
6.23	Regulator output . . . . .	94
6.24	System power solded to the Printed Circuit Board (PCB) . . . . .	94
7.1	State diagram of the Testbed . . . . .	97
7.2	Block diagram of the software . . . . .	98
7.3	Block diagram of the LCD task . . . . .	102
7.4	Shake game screen . . . . .	103
7.5	Finding state screen . . . . .	103
7.6	Found state screen . . . . .	103
7.7	Block diagram of the Read Sensor task . . . . .	104
7.8	Block diagram of the light game . . . . .	105
7.9	Position of the LDR respect to 0° . . . . .	107
7.10	Block diagram of the shake game . . . . .	108
7.11	Step response for X axis motor . . . . .	110
7.12	Step response for Y axis motor . . . . .	111
7.13	Step response for Z axis motor . . . . .	112

7.14 Rise time for X axis motor . . . . .	113
7.15 Rise time for Y axis motor . . . . .	113
7.16 Rise time for Z axis motor . . . . .	114
7.17 $1\ \Omega$ resistance . . . . .	115
7.18 $1\ \Omega$ resistance soldered to the PCB . . . . .	115
7.19 Capture of the oscilloscope screen . . . . .	116
7.20 Extended capture of the oscilloscope screen . . . . .	117
7.21 Time taken to read the sensors . . . . .	119
7.22 Time taken to do the PID calculations . . . . .	120
7.23 Block diagram of the serial communication task . . . . .	121
7.24 Block diagram of the serial interrupt task . . . . .	122
B.1 PART PER MILLION (PPM) measurement process with Kalibrate-RTL . . .	139



# LIST OF VIDEOS

4.1 Gyroscope use example . . . . .	39
6.1 3D view of the PCB . . . . .	88
7.1 “GoodBye gesture” . . . . .	106
8.1 Shake game video . . . . .	123
8.2 Light game video . . . . .	124



# LIST OF TABLES

4.1	CP2525 air compressor characteristics . . . . .	27
4.2	Variation of the friction-less scenary temp with the weight and the air pressure	28
4.3	Variation of the friction-less scenary time with the air pressure for 1,25 kg . . . . .	29
4.4	ATmega328 main features [40] . . . . .	33
4.5	Prices for step up and charger module . . . . .	35
4.6	MPU6050 general features [40] . . . . .	37
4.7	MPU6050 gyroscope features [40] . . . . .	37
4.8	MPU6050 accelerometer features [40] . . . . .	38
4.9	LSM303 features [49] . . . . .	43
4.10	Scaling factor for LSM303 accelerometer [49] . . . . .	43
4.11	Scaling factor for LSM303 magnetometer [49] . . . . .	43
4.12	Minimun and maximun values for light sensors . . . . .	45
4.13	DT-2234B photo tachometer specifications [43] . . . . .	47
4.14	Minimun PWM value neccessary b each axis wheel to start to turn . . . . .	47
4.15	Correspondence between voltage and PWM pulse in X axis motor . . . . .	48



4.16 RPM X axis motor forward . . . . .	49
4.17 RPM X axis motor backward . . . . .	50
4.18 Correspondence between voltage and PWM pulse in Y axis motor . . . . .	51
4.19 RPM Y axis motor forward . . . . .	52
4.20 RPM Y axis motor backward . . . . .	53
4.21 Correspondence between voltage and PWM pulse in Z axis motor . . . . .	54
4.22 RPM Z axis motor forward . . . . .	55
4.23 RPM Z axis motor backward . . . . .	56
4.24 Xbee specifications [26] . . . . .	57
4.25 Zieger-Nichols parameters [35] . . . . .	61
6.1 ENIX MGL2803 Battery Specifications [27] . . . . .	76
6.2 Indicator light state [24] . . . . .	91
7.1 FreeRTOS main features [12] . . . . .	100
7.2 FreeRTOS memory usage [13] . . . . .	101
7.3 Information displayed for each state . . . . .	104
7.4 Step response for X axis motor . . . . .	109
7.5 Step response for Y axis motor . . . . .	111
7.6 Step response for Z axis motor . . . . .	112
7.7 Rise time for each motor . . . . .	114
7.8 Needed data to calculate the Zieger Nichols parameters . . . . .	117
7.9 Results of Zieger-Nichols parameters for X axis . . . . .	117
7.10 Results of Zieger-Nichols parameters for Y axis . . . . .	118
7.11 Results of Zieger-Nichols parameters for Z axis . . . . .	118
7.12 Results of Zieger-Nichols parameters for X axis . . . . .	118
7.13 Results of Zieger-Nichols parameters for Y axis . . . . .	118
7.14 Results of Zieger-Nichols parameters for Z axis . . . . .	118
7.15 Minimum and maximum sample time for each axis . . . . .	120

7.16	Commands to communicate with the Testbed . . . . .	121
A.1	Optiboot features [6] . . . . .	133
A.2	AVRdude command options [6] . . . . .	134
B.1	Kalibrate command options . . . . .	138
C.1	Power Supply building cost . . . . .	141
C.2	Other electronics devices cost . . . . .	142
C.3	Total cost of the PCB implementation . . . . .	142
C.4	Software cost . . . . .	142
C.5	Junior engineer cost . . . . .	143
C.6	Senior engineer cost . . . . .	143
C.7	Total human resources cost . . . . .	143



# GLOSSARY

**ADCS** Attitude Determination and Control System. It is used to regulate, stabilize and orient the [Cubesat](#).

**Air-bearing** Element of the [Testbed](#) that ejects air under pressure through some small holes placed on a surface, normally spherical, to generate an air layer between the air-bearing and the rotor of the Testbed platform, creating a friction-less scenario.

**Altium Designer** Altium Designer is an electronic design automation software package for printed circuit board, FPGA and embedded software design, and associated library and release management automation. It is developed and marketed by Altium Limited of Australia. The current release is 16.0. [\[50\]](#).

**Arduino** is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs and turn it into an output [\[10\]](#).

**Bootloader** Is a piece of firmware in the microcontroller that allows installing new firmware using an external programmer [\[2\]](#).

**Buzzer** Is a piece of firmware in the microcontroller that allows installing new firmware using an external programmer [\[2\]](#).

**Cubesat** Is a type of miniaturized satellite for space research that have made by modules with a volume of one litre and a mass of no more than 1.33 kilograms. Often use commercial components for their electronics and structure [\[52\]](#). The simplest Cubesat is the 1U Cubesat, that only has one of this modules.

**FUNcube Dongle** Is a software-defined radio (SDR) covering all broadcast and amateur radio bands from 150kHz to 1.9GHz. With an SDR, many of the functions of a traditional hardware-only radio are performed in software, such as demodulation, decoding and frequency conversion.

**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 <https://granosat.ugr.es/index.php/es/>.

**Kalibrate-RTL** Software that calculate the ppm offset by using the precision clock of a GSM mobile phone base station.

**KPIB** Is a framework for operating laboratory instruments that are connected to a computer by GPIB or serial connections. KPIB provides an unified interface for communicating with different instruments of the same type from different manufacturers. KPIB requires the MATLAB Instrument Control toolbox.

**LM1117** The LM1117 is a low dropout voltage regulator manufactured by Texas Instruments with a dropout of 1.2 V at 800 mA of load current. It is available in five fixed voltages, 1.8 V, 2.5 V, 3.3 V, and 5 V and it offers current limiting and thermal shutdown [37].

**MAX1674** It's a compact, high-efficiency, step-up DC-DC converters fit in small  $\mu$ MAX packages [38].

**MAX8211** It's a CMOS micropower voltage detector that warns microprocessors ( $\mu$ Ps) of power failures [39].

**MGL2803** Lithium-ion battery from Enix Energies with 3.75 V nominal voltage [27].

**MT3608** The MT3608 is a constant frequency, 6-pin SOT23 current mode step-up converter up to 28V output voltage intended for small, low power applications [15].

**Proccesing** Is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology [46].

**RTL-SDR** Is a very cheap software defined radio that uses a DVB-T TV tuner dongle based on the RTL2832U chipset. With the combined efforts of Antti Palosaari, Eric Fry and Osmocom it was found that the signal I/Q data could be accessed directly, which allowed the DVB-T TV tuner to be converted into a wideband software defined radio via a new software driver.

**SDR#** Is probably the most popular software program that is used with the [Software Defined Radio \(SDR\)](#) devices like [FUNcube Dongle](#) and [RTL-SDR](#). It is free, fast and fairly easy to use.

**Testbed** A Testbed (also “test bed”) is a platform for conducting rigorous, transparent and replicable testing of scientific theories, computational tools and new technologies [55].

**TP4056** Is a complete constant-current/constant-voltage linear charger for single cell lithium-ion batteries [24].

**USBtinyISP** Is an open-source USB AVR programmer and [SPI](#) interface. It is low cost, easy to make, works great with avrdude, is AVRStudio-compatible and tested under Windows, Linux and MacOS X. Perfect for students and beginners, or as a backup programmer [7].

**XBee** XBee is the brand name of a family of form factor compatible radio modules from Digi International [56].



# ACRONYMS

**ADC** Analogue to Digital Converter.

**AGC** Automatic Gain Control.

**AM** Amplitude Modulation.

**CW** Continuous Wave.

**DMP** Digital Motion Processor.

**DSB** Double Side Band.

**DVB-T** Digital Video Broadcasting-Terrestrial.

**ECTD** Electronics and Computer Technology Department.

**FFT** Fast Fourier Transform.

**FM** Frequency Modulation.

**GPIB** General Purpose Interface Bus.

**GSM** Global System for Mobile communications.

**HMI** Human-Machine Interface.

**ICSP** In Circuit Serial Programming.



**IDE** Integrated Development Environment.

**IF** Intermediate Frequency.

**LCD** Liquid Crystal Display.

**LDR** light-Dependent Resistor.

**LED** Light-Emitting Diode.

**LNA** Low Noise Amplifier.

**LSB** Low Side Band.

**MATLAB** [MATrix LABoratory](#).

**MEMS** Microelectromechanical Systems.

**NFM** Narrow Frequency Modulation.

**NOAA** National Oceanic and Atmospheric Administration.

**OBC** On Board Controller.

**PCB** Printed Circuit Board.

**PID** Proportional Integral Derivative.

**ppb** Part Per Billion.

**ppm** Part Per Million.

**PWM** Pulse Width Modulation.

**RPM** Revolutions Per Minute.

**RTOS** Real Time Operating System.

**SAW** Surface Acoustic Wave.

**SDR** Software Defined Radio.

**SINAD** Signal to Noise And Distorsion ratio.

**SMA** SubMiniature version A.

**SMD** Surface Mount Devices.

**SPI** Serial Peripheral Interface.

**TCXO** Temperature Compensated Crystal Oscillator.

**UGR** University of Granada.

**USB** Universal Serial Bus.

**USB** Upper Side Band.

**WFM** Wide Frequency Modulation.



## CHAPTER

# 1

# INTRODUCTION

## 1.1 Objectives and motivation

The present dissertation completes the studies of the master in telecommunications engineering. When I was searching for a theme for doing the final project, I wanted to work in something that really completed my education in the university. Before I studied the telecommunications master, I studied the telecommunications technology degree with mentions in telematics in the University of Granada. After obtaining the college degree and passing all the obligatory subjects of the master, I believed that, although I had some knowledge in telematics and signal processing, I had very little knowledge in other disciplines like electronics. For this reason I wanted to do a final project that helped me to learn about subjects that I had not learnt enough during the master and the degree.

I had a little experience doing dissertations thanks to my final project degree. In my case, it consisted on a simulation of a network attack in MANET<sup>1</sup> networks. Although it was very interesting and allowed me to learn a lot about network attacks, I did not want to do a simulation again neither something similar. I wanted to do something that I could touch with my own hands, close to the real working world than the theoretical investigation.

[GranaSAT](#), whose logo is shown in figure 1.1, is an aerospace project of the University of Granada, whose main objective is to design and build a [Cubesat](#) by students [32]. This project is coordinated by the professor Andres Maria Roldan Aranda and it allows the students to obtain knowledge about the aerospace field and a real experience in this area.

---

<sup>1</sup>Mobile ad hoc network



**Figure 1.1** – Logo *GranaSAT*

*GranaSAT* offered me what I was looking for and, when I had the opportunity of doing my final project in *GranaSAT*, I accepted it without doubting. Although I did not know anything about aerospace, when I began to read documentation, I started to like it quickly and, in the future, I would like to work in this field.

## 1.2 Cubesat background

As it was said before, the main objective of the *GranaSAT* project it to design and build a *Cubesat*. It is a type of miniaturized satellite for space research that is made by modules with a volume of one litre and a mass of no more than 1.33 kilograms. It often is used commercial components for their electronics and structure [52]. Figure 1.2 shows an example of *Cubesat*, while figure 1.3 shows differents dimensions for *Cubesat*. The simplest is the 1U *Cubesat*, that is made by only one cube of 10x10x10 cm and is the kind of *Cubesat* that is being designed in *GranaSAT*.



**Figure 1.2** – *Ncube-2*, a Norwegian *Cubesat* [52]

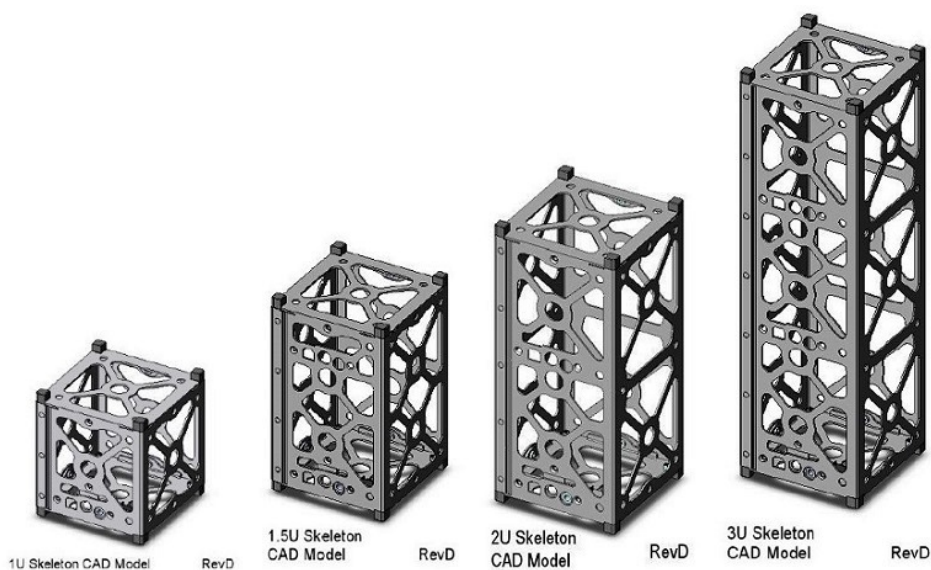


Figure 1.3 – Different dimensions for Cubesats [22]

Figure 1.4 shows an example of the block diagram of a Cubesat, where we can see the main parts and their connections with the OBC. One block of the diagram is the ADCS block. This module, which will be described deeply in 1.2.1, is used to regulate, stabilize and orient the Cubesat. An important percentage of the present project is dedicated to develop this module as it will be described in section 2.1.

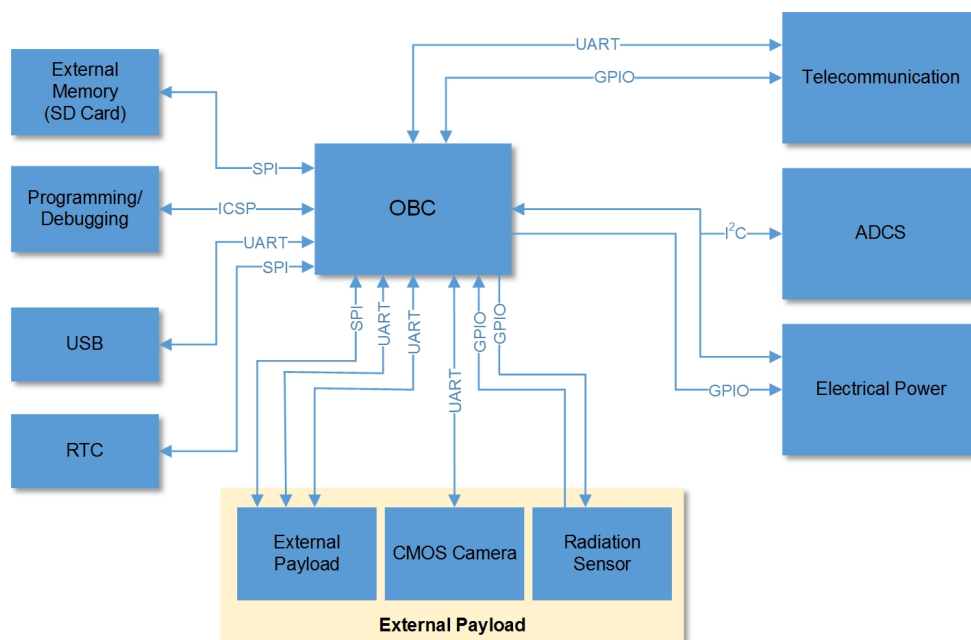


Figure 1.4 – Cubesat block diagram [25]

### 1.2.1 Attitude Determination and Control System

The ADCS regulates, stabilizes and orients the Cubesat. The attitude of a satellite is the orientation of the spacecraft body respect with a defined external reference. The ADCS has sensors like gyroscopes and other inertial sensors that allow to determine the orientation of the Cubesat. The attitude control is the maintenance of a desired, specified attitude within a given tolerance. In order to control the attitude, the ADCS has actuators that generates torques that moves the Cubesat to the desired attitude. Figure 1.5 shows a better way to understand the ADCS [21].

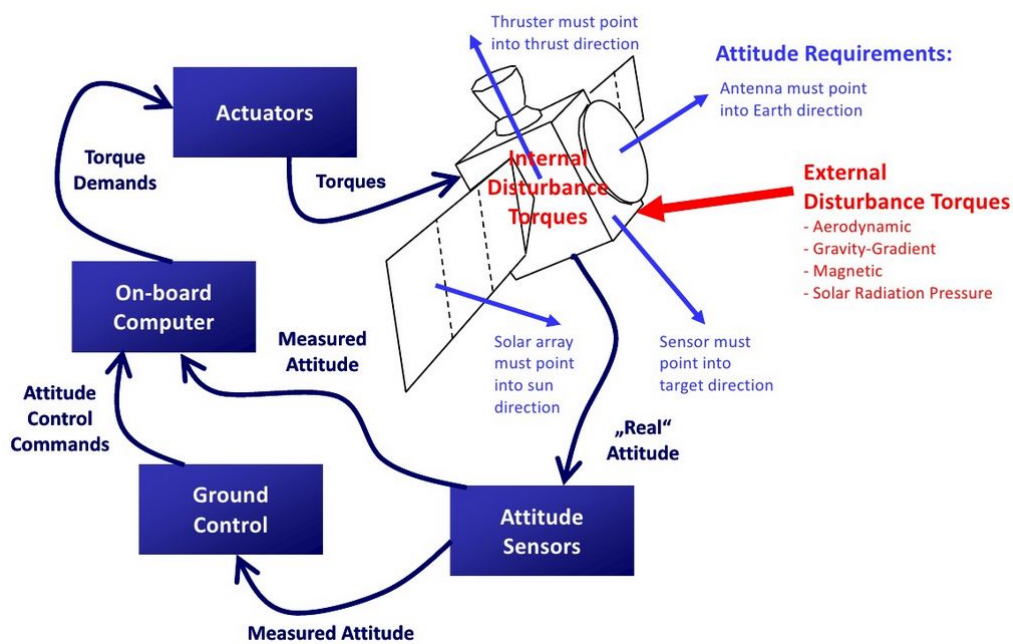


Figure 1.5 – ADCS for a satellite [21]

There are three different actuators that can be used in a satellite to control the attitude. One of the most common are the thrusters, but they required a significant quantity of fuel for their functionality. This is a relevant constraint for a Cubesat, because of its size and its limited weight.

Other example of actuators are the magnetorquers. They work creating a magnetic field against the Earth local magnetic field, generating a torque that moves the Cubesat.

Besides the thrusters and magnetorquers, the third kind of actuators are the reaction wheels. This actuator consists in massive wheels that, when they rotate, it is generated an angular momentum that produces inertial torques.

In some Cubesat have both systems installed, magnetorquers and reaction wheels <sup>2</sup> (see

<sup>2</sup>Thrusters are not used in Cubesat because the size and weight constraint.

figure 1.6), because the external disturbances (produced for example by solar radiation, magnetic field from the Earth, gravity gradient torque, etc), generates angular momentum within the Cubesat, and the reaction wheels store the angular momentum, but not dissipate it. When the reaction wheels reach their maximum storage of angular momentum, external torques, in our case, magnetorquers, reduced it (desaturating the reaction wheels). Furthermore, the Earth's magnetic field decreases proportional with distance, for that reason, the magnetorquers are useful only for low Earth orbits [21].

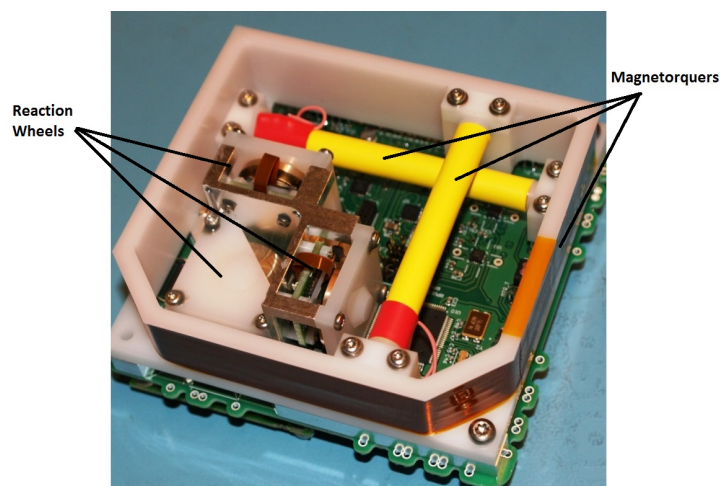


Figure 1.6 – Cubesat Delfi-n3Xt actuators

Figure 1.7 shows the block diagram of the ADCS module that will be included in the GranaSAT Cubesat.

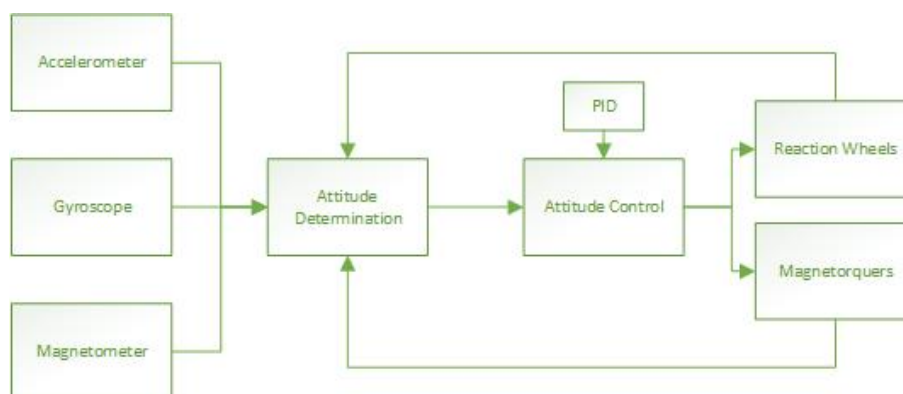


Figure 1.7 – ADCS GranaSAT block diagram [21]

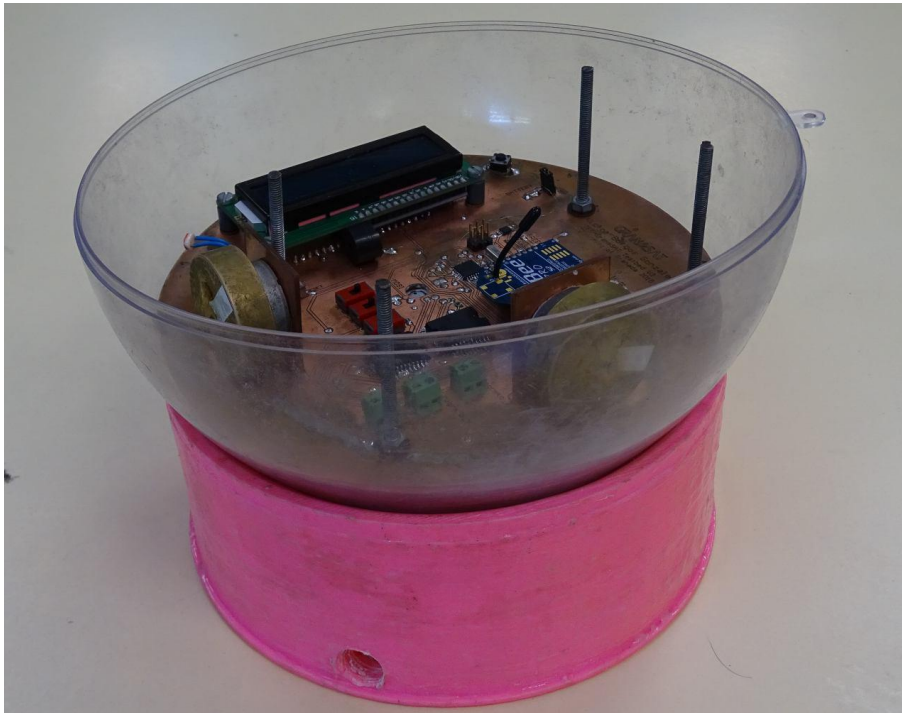


## 1.2.2 Testbed

In order to test all over the systems for the **Cubesat**, a **Testbed** is needed. A **Testbed** is a simulation platform designed to test an aerospace system, in this case, the **GranaSAT** subsystems. The testbed imitates the friction-less environment of the space scenario and it takes measurements with sensors to analyze this scenario.

There are several methods to create the friction-less scenario, but the most common in the aerospace field is using an air-bearing. This solution cannot create a no-gravity environment, but it has a nearly free torque condition. The function of that air-bearing is to eject air under pressure through some small holes placed on a surface, normally spherical, to generate an air layer between the **air-bearing** and the rotor of the **Testbed** platform [21].

Victor Burgos Gonzalez designed and built a **Testbed** in his degree thesis (figure 1.8) [21] that allows to test the **ADCS** module.



**Figure 1.8** – *Testbed designed and built by Victor Burgos*

## CHAPTER

# 2

# CLIENT REQUESTS AND REQUIREMENTS

## 2.1 Client requests

In a first approach, it was proposed several objectives by the client (the project's tuthor in this case) that must be achieved to complete successfully the present project. These objectives will now be described:

- **Objective 1.** The [SDR](#) systems are being using in [GranaSAT](#) project as radiocommunications receptors for the reception of signals from satellites like [National Oceanic and Atmospheric Administration \(NOAA\)](#) meteorological satellites or others [Cubesat](#) launched by other companies (in the future perhaps we will be able to receive signals from our own [Cubesat](#)). In order to optimizate the signal reception, it is necessary to analyze the performance of our [SDR](#) systems.
- **Objective 2.** As it was said in section [1.2.2](#), Victor Burgos Gonzalez designed and built a [Testbed](#) in his bachelor thesis [[21](#)]. Although the designed [PCB](#) was correctly implemented and the electronic works well, it only was programmed for tested the differents sensors of the [Testbed](#). The present project is thought to continue the work made by Victor programming the microprocessor to works as a [ADCS](#) module. Once the [Testbed](#) will be completely finished, it is thought to show it to high school students in the Science Week of the science faculty, in order to attract them to engineering

studies. For this reason it is necessary to add an operating mode that seems visually appealing for young students from high school. Specifically, it is thought to program the [Testbed](#) as a light detector so that the [Testbed](#) looks for a light source and orients his sensors to the source.

## 2.2 Requirements

Once the clients requests are known, we have to know what is necessary in technical terms to achieve the aims successfully. In this section, the requirements needed for each one of the objectives mentioned in [2.1](#) will be described.

### 2.2.1 Requirements for performance analysis of the SDR

A [SDR](#) can be described as a radiocommunications system where most of components are implemented by software instead of hardware, using a personal computer or an embeded computer device. Although during the Master Degree we have studied the [SDR](#) systems theoretically, we never have used one of this systems. So the first step will to be familiarized with this hardware, as well as the software needed to use it in the computer. In order to measure the technical characteristics of the [SDR](#), we will have to use laboratory devices like communication test set and signal sources, so it is necessary to be familiarized with these devices too.

Finally, we have to decide about what parameter of the [SDR](#) we have to measure their performance and design an appropriated measurement process.

### 2.2.2 Requirements for programming the ADCS

This work will be a continuation of the work made by Victor Burgos Gonzales in his bachelor thesis [\[21\]](#), so the first step is to understand all the desing process of the [Testbed](#) and how it works. Moreover, the [Testbed](#) was designed and built one year before the present project was started so, it is possible that some aspects of the electronics do not work correctly and we will have to repair them.

The [Testbed](#) designed by Victor was thought to work as a mobile device so that it was not necessary that the [Testbed](#) was fixed to a power source. However, a mistake in the design of the [Testbed](#) power supply made this impossible. It will be designed and implemented a new power supply in order to the [Testbed](#) becomes mobile.

An importat element of the [Testbed](#) is the [air-bearing](#) that allows us simulate the friction-less scenario. We have to ensure that the [air-bearing](#) built by Victor, gives us a friction-less scenario and improves it if it will be possible. On the other hand, the effectiveness of the [air-bearing](#) depends on the air compressor. The air compressor available in the laboratory that Victor used for his tests had a little deposit, so his experiments cannot last a long

time. It would be desirable that we could find a new air compressor that allows us to do experiments for a longer time.

Another aspect to take into account is the communication with the [Testbed](#) that will be necessary for the debugging while the [Testbed](#) is being programmed as well as sending data to the ground station when the [ADCS](#) module will be implemented. The [PCB](#) has a [XBee](#) module that allows to send information by serial port using Zigbee technology. We will program the microprocessor so that allows us the communication in this way.

The programming of the [ADCS](#) requires a deep knowledge about the behavior of the sensors and actuators in a friction-less scenario so, we will have to understand how these sensors (gyroscope, accelerometer, magnetometer and light sensors) works and how to use them in our particular environment. Moreover, as the actuators are going to make the [Testbed](#) turn around itself, we need to know the physics about the inertial momentum.

We cannot forget that our [Testbed](#) is thought for an aerospace application. These kind of applications tend to be very expensive and need to guarantee a right working free from mistakes. For this reason, it would be desirable to program the [Testbed](#) using a [Real Time Operating System \(RTOS\)](#) that assure us concrete time periods in the program execution and avoid that the system freezes or that the program go into an infinite loop.

To program the [Testbed](#) as a light detector, we need light sensors. However, despite the [PCB](#) has been designed with the needed routes to adding these sensors, the sensors are not included so we have to add it. Moreover, it would be a good idea to add another mode of operation apart from the light detector.

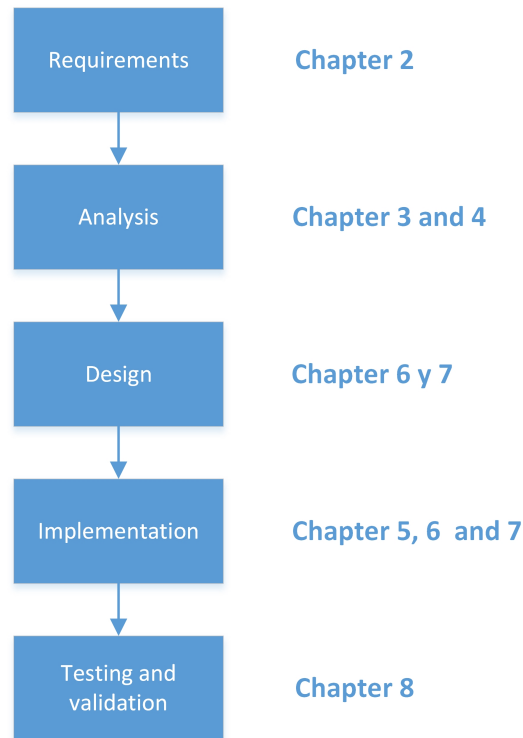
Finally, as it was said in [2.1](#), our [Testbed](#) must be visually attractive so we have to considerer to add some devices like [LEDs](#) or a buzzer that give us visual information without having to watch the [LCD](#) screen.

## 2.3 Project structure

Once the requirements have been exposed, we have to take decisions about the different alternatives in which we can implement these requirements, as well as making a planning that allows us to conclude this project successfully. Moreover, an important part of this project is based in the hardware made by Victor Burgos in his degree thesis so, it is necessary to analyze the whole system that he built.

In this project we have the particularity that we are working in two very different topics. By one hand, we are going to analyze the [SDR](#) systems and, on the other hand, we are going to program the [Testbed](#). For both tasks, we have follow the analysis, design and implementation process. In order to prevent confusions in the reader, in figure [2.1](#) is shown a diagram of the engineering process accompanied by the related chapter and the related task. First we will explain the analysis process of the [SDR](#) systems (chapter [3](#)) and of the [Testbed](#) (chapter [4](#)). Then we have to talk about the design and implementation process.

As we have two very different tasks, in order to get a better understanding, first we will talk about the desing, implementation a results of the [SDR](#) task (chapter 5) and then we will talk about the desing, implementation a results of the [Testbed](#) task (chapters 6, 7 and 8).[at the end of the section](#) is shown how the schedule of this project has been flown with a Gant diagram.



**Figure 2.1** – *Project structure*

	Nombre	Duracion	Inicio	Terminado	Tri 4, 2015			Tri 1, 2016			Tri 2, 2016			Tri 3, 2016			T		
					oct	nov	dic	ene	feb	mar	abr	may	jun	jul	ago	sep		o	
1	<b>SDR accuracy measurement</b>	<b>71 days</b>	<b>21/09/15 9:00</b>	<b>29/12/15 9:00</b>															
2	SDR analysis	30 days	21/09/15 9:00	2/11/15 9:00															
3	PPM measurement	11 days	2/11/15 9:00	17/11/15 9:00															
4	Optimizating signal reception	10 days	17/11/15 9:00	1/12/15 9:00															
5	Accuracy measurement	20 days	1/12/15 9:00	29/12/15 9:00															
6	<b>Testbed programming</b>	<b>150 days</b>	<b>16/11/15 9:00</b>	<b>13/06/16 9:00</b>															
7	PCB analysis	94 days	23/11/15 9:00	1/04/16 9:00															
8	Air-bearing analysis	15 days	16/11/15 9:00	7/12/15 9:00															
9	<b>Power supply design and imple...</b>	<b>75 days</b>	<b>23/11/15 9:00</b>	<b>7/03/16 9:00</b>															
10	Power supply design	40 days	23/11/15 9:00	18/01/16 9:00															
11	Power supply implementation	10 days	22/02/16 9:00	7/03/16 9:00															
12	Testbed programming	51 days	1/04/16 9:00	13/06/16 9:00															
13	Testing	32 days	13/06/16 9:00	27/07/16 9:00															
14	Documentation	117 days	30/03/16 9:00	9/09/16 9:00															

2

## CHAPTER

# 3

# SDR ANALYSIS

A [SDR](#) can be described as a radiocommunications system where most of components are implemented by software instead of hardware, using a personal computer or an embebed computer device. The main advantage of these systems is that it is possible to use a general purpose microprocessor for the signal processing, which decrease the total cost of a radiocommunications system. Furthermore, a software defined radio system is more flexible due to the software permits a wider range of possibilities in the configuration. All the [SDR](#) systems are composed of two main elements: a hardware device that receives the signal and a software which configures the device (setting the demodulation mode, the gain and the frequency band).

[SDR](#) devices are being using in the [GranaSAT](#) project for the reception of signals from [NOAA](#) meteorological satellites. For this task, we have two different [SDR](#) devices: the [RTL-SDR](#) and the [FUNcube Dongle](#). In section [3.1](#) and [3.2](#) these devices will be described in depth as well as the software used in section [3.3](#).

### 3.1 RTL-SDR

By using custom software drivers, a commonly used cheap TV dongle can be turned into a sophisticated [SDR](#) with similar features to others high price devices. The [RTL-SDR](#) is an extremely cheap [SDR](#) which is based on [DVB-T TV USB](#) receiver dongles that have the [RTL2832U](#) chip in them. Of course, the performace of these dongles will not be equal to a dedicated [SDR](#), but they perform extemely well for the price and make it perfect for the



GranaSAT project [42].

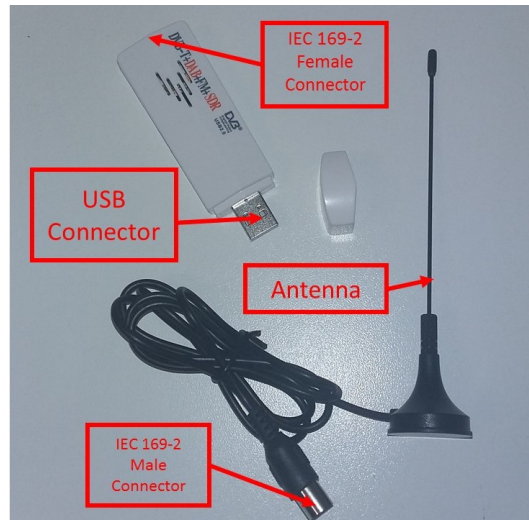


Figure 3.1 – *RTL-SDR* used in *GranaSAT* project

Picture 3.2 shows a basic block diagram of the *RTL-SDR* which basically is composed by a R820T tuner and a *ADC*. All the signal processing is done by software.

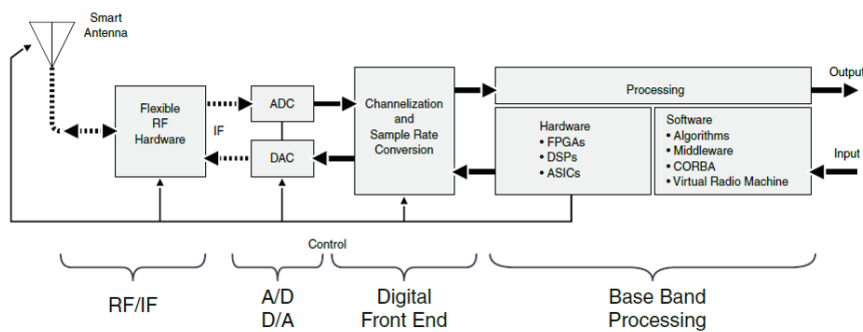


Figure 3.2 – *RTL-SDR* block diagram

*RTL-SDR* technical specifications[42]:

- 22-2200 MHz tunable range
- 3.2 MHz max bandwidth (2.8 MHz stable)
- 8-bit *ADC* giving 50 dB dynamic range
- < 4.5 dB noise figure *LNA*
- 75 Ohm input impedance

## 3.2 FUNcube Dongle

The [FUNcube Dongle](#) hardware is designed to be very simple to set up, with the only connections being an antenna connection and a USB connection. There are no physical controls on the [FUNcube Dongle](#) itself: all the settings for the [FUNcube Dongle](#) are controlled from host computer. The [FUNcube Dongle](#) is very similar to the [RTL-SDR](#) device. The main difference is that the first one has an oscillator with a high quality quartz crystal and this permits to obtain a very high accuracy [30]. According to the technical specifications described below, the crystal oscillator has a 1.5 ppm precision. This means that for each 1 MHz, the tune only has 1.5 Hz of deviation.



Figure 3.3 – *FUNcube Dongle* used in *GranaSAT* project

[FUNcube Dongle](#) technical specifications [30]:

- Guaranteed frequency range: 150 kHz to 240 MHz and 420 MHz to 1.9 GHz
- Typical frequency coverage: 150 kHz to 260 MHz and 410 MHz to 2.05 GHz
- [TCXO](#) specified at 0.5 ppm (in practice about 1.5 ppm)
- 192 kHz sampling rate
- Extremely simple hardware setup, just two connections:
  - Standard [SMA](#) female antenna port for input
  - [USB](#) 1.x type A male connection for output
- Eleven discrete hardware front end filters including:
  - 6 MHz 3 dB bandwidth (10 MHz at -40 dB) [SAW](#) filter for the 2 m band
  - 20 MHz 3 dB bandwidth (42 MHz at -40 dB) [SAW](#) filter for the 70 cm band

- Third- and fifth-order LC bandpass filters for other bands
- Front end [LNA](#) OIP3 30 dB.
- Typical noise figures:
  - 50 MHz 2.5 dB
  - 145 MHz 3.5 dB
  - 435 MHz 3.5 dB
  - 1296 MHz 5.5 dB
- Typical [NFM](#) 12 dB [SINAD](#) measurements:
  - 145 MHz 0.15  $\mu$ V
  - 435 MHz 0.15  $\mu$ V
- No additional drivers required for Linux, OSX or Windows
- Integrated 5 V bias T switchable from software

## 3

### 3.3 SDR Software. [SDR#](#)

As it has been said before, a [SDR](#) needs a software for the set up. It is possible to find a lot of software for this purpose, most of them free. [SDR#](#) (SDRSharp) is probably the most popular software program that is used with the [SDR](#) devices like [FUNcube Dongle](#) and [RTL-SDR](#). It is free, fast and fairly easy to use. Furthermore, there are a lot of information and tutorials available and it is compatible with both [SDR](#) devices. For these reasons we have chosen [SDR#](#) for use it in this project.

[SDR#](#) allows:

- To tune the [SDR](#) device to the desired frequency
- To choose the demodulation mode ([NFM](#), [WFM](#), [AM](#), [DSB](#), [LSB](#), [USB](#) and [CW](#))
- To obtain the demodulated signal as a sound signal and to hear it
- Provides [AGC](#)
- To display the [FFT](#) signal
- To reduce the audio and the [IF](#) noise
- To record the baseband and audio signal

The picture 3.4 shows the main view of the [SDR#](#) software. In this example it can be seen a range (between 97,250 MHz and 99,750 MHz) of the [FM](#) commercial radio band. Each lobe corresponds to a radio station broadcasting in a different frequency. Below, we can see the energetic level of the signal (red color means high energetic level) along the time and the frequency and, finally, on the left there are the configuration and set up options.

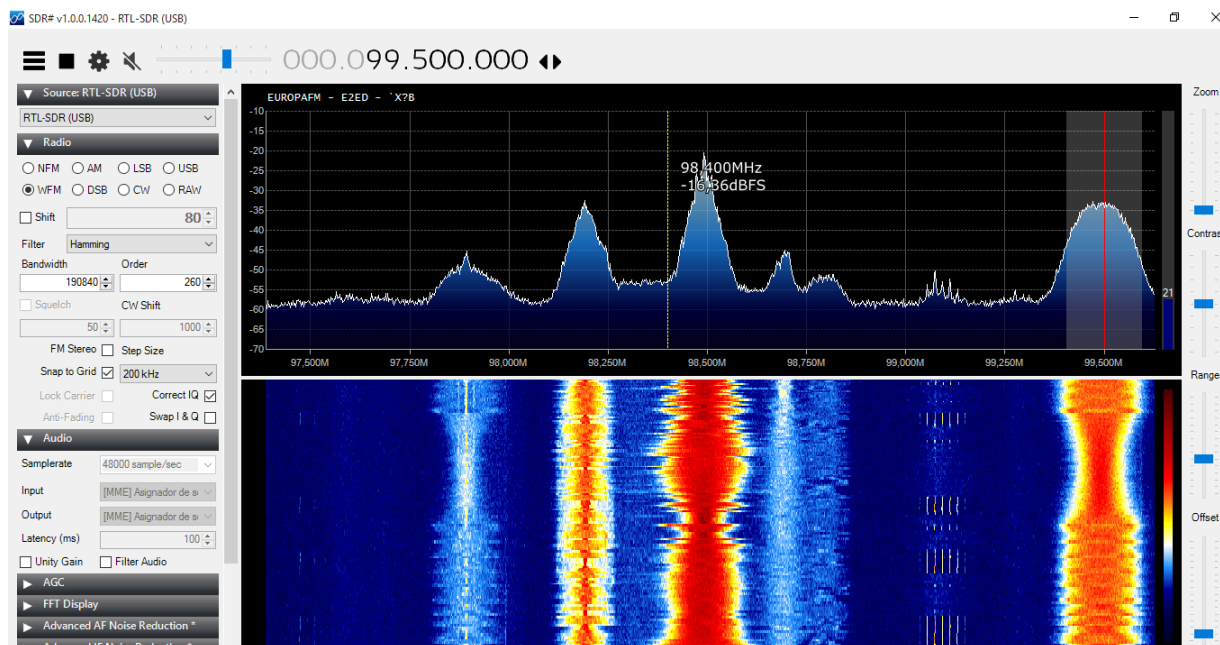


Figure 3.4 – Main view of [SDR#](#)

### 3.4 SDR parameters

In sections 3.1 and 3.2 the lector can observe the technical parameters of the [RTL-SDR](#) and [FUNcube Dongle](#) respectively given by the manufacturer. Although [FUNcube Dongle](#) manufacturer gives us more information than [RTL-SDR](#) manufacturer, we cannot see high differences between them. However, [FUNcube Dongle](#) is much more expensive than the second one. We can found the reason in [42, pág 28], where we can read that [RTL-SDR](#) has a low quality 28.8 MHz crystal oscillator and the tuned frequency can be out by approximately  $\pm 100$  ppm offset deviation. This means that for each 1 MHz tuning in frequency, the signal will be moved 100 Hz. For example, if we tune to 100 Mhz with a [RTL-SDR](#) with a 64 ppm offset, the final tuning will be 100,064 MHz. On the other hand, in 3.2 we can read that [FUNcube Dongle](#) manufacturer guarantees us a offset deviation of 1,5 ppm, so we can deduce that [FUNcube Dongle](#) has a higher quality crystal oscillator that is able to tune with a lower deviation. At this point, we have to ask ourselves how the quality of the crystal oscillator affects the signal reception.

In picture 3.4 we can see a signal reception done with the [RTL-SDR](#) and it can be observed

that, although the tune is set to 99,5 MHz, the tune is not situated exactly in the center of the lobe. Another example can be seen in picture 3.5, where a 300 MHz signal is input in the SDR and the tune does not focus in the center of the lobe again.

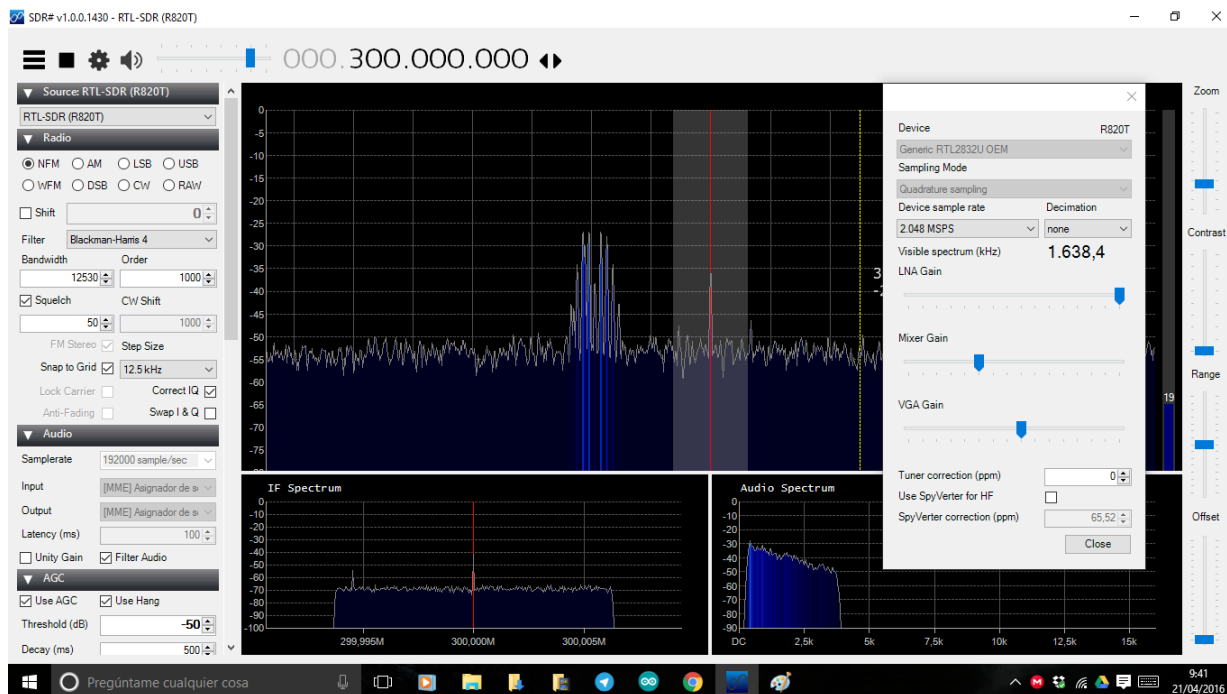


Figure 3.5 – *RTL-SDR tuning without ppm correction.*

Now we are going to receive a signal with the [FUNcube Dongle](#), which we can see in picture 3.6. In this case, the signal is exactly centered in the middle of the lobe (after introducing the 1,5 offset correction given by the manufacturer) so we can deduce that although both the devices allow us the demodulation and decoding of signals in the amateur radio bands, the [FUNcube Dongle](#) is much precise in the tune and, consequently, is more expensive. However, although the [RTL-SDR](#) has a high deviation in his tune, this deviation can be corrected as it will be explained in section 5.1.1.

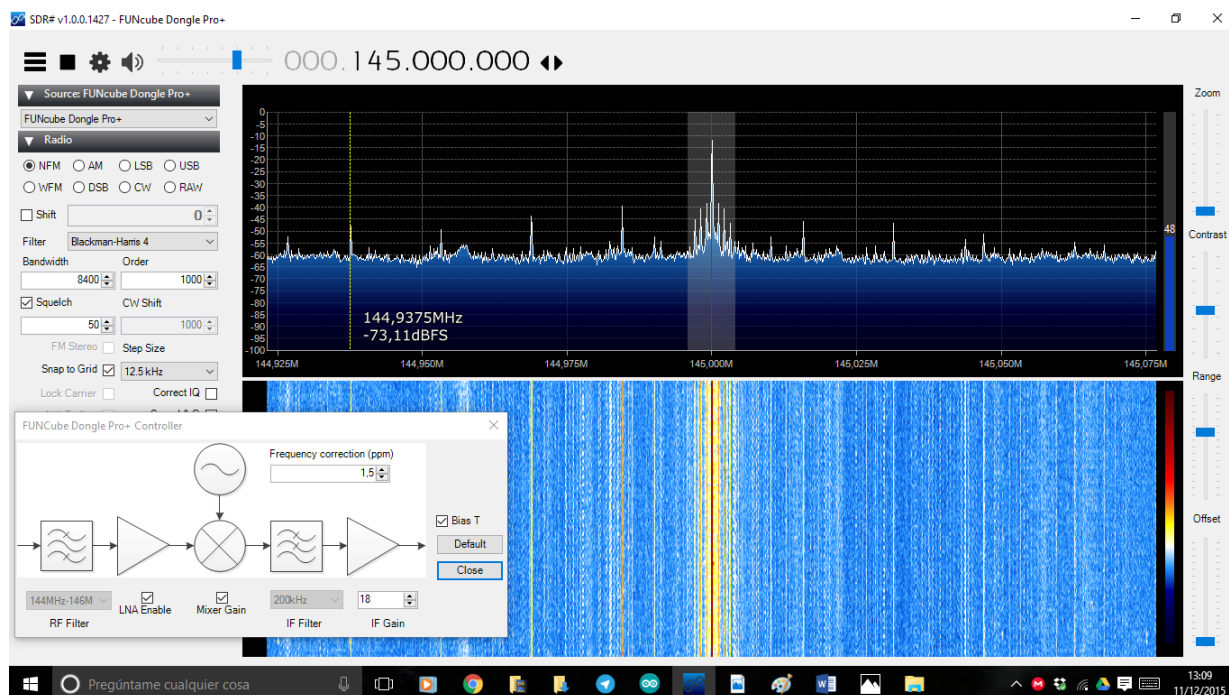


Figure 3.6 – *FUNcube Dongle* tuning with ppm correction.

Other question to take into account is how much energy level needs the SDR system to receive a signal correctly. This is important because if a device is able to receive correctly a signal with a very low energy level, this means that the device has a high sensibility. *FUNcube Dongle* technical specifications (3.2) says that the device needs  $0,15 \mu\text{V}$  (-124 dBm) to reach 12 dB SINAD at 145 MHz and 435 MHz. In the case of *RTL-SDR* we cannot find any information about how much energy level is required in his technical specifications (3.2) neither in his main website ([47]). Moreover we tried to get in touch with the *RTL-SDR* manufacturers to ask them about this question but we did not obtain any answer. However, we can suppose that *RTL-SDR* will need more energy level to reach 12 dB SINAD because the quality of his crystal oscillator is higher. Furthermore, since *FUNcube Dongle* manufacturer gives us the energy level needed for two specific frequencies (145 MHz and 435 MHz), we can suppose that the needed energy level changes with the frequency. The needed energy level in the *RTL-SDR* would change too because of the low quality of his oscillator.

To sum up, the quality of a SDR system depends specially on his crystal oscillator, that give us more or less precision in the tune and allows to receive signals with a certain accuracy. The deviation in the tune can be easily calculated and corrected for both *RTL-SDR* and *FUNcube Dongle*. However, we have not enough information about the sensibility of these devices and how it changes with the frequency. For this reason, we are going to measure how the sensibility of the SDR changes with the frequency (chapter 5). However, first we have to decide a method of measuring, which will be explained in section 3.5.

### 3.5 Accuracy measurement method

In order to measure the SDR sensibility, we have to find the minimum energy level for the signal that allows us to receive the signal correctly in the audio receptor. We are going to need a signal generator that produces the signal that is received by the SDR, and an audio receptor that receives the signal demodulated by the SDR. In the laboratory we have available the Marconi 2022 signal generator (3.7) and the HP 8903B audio analyzer (3.8).



Figure 3.7 – Marconi 2022 signal generator



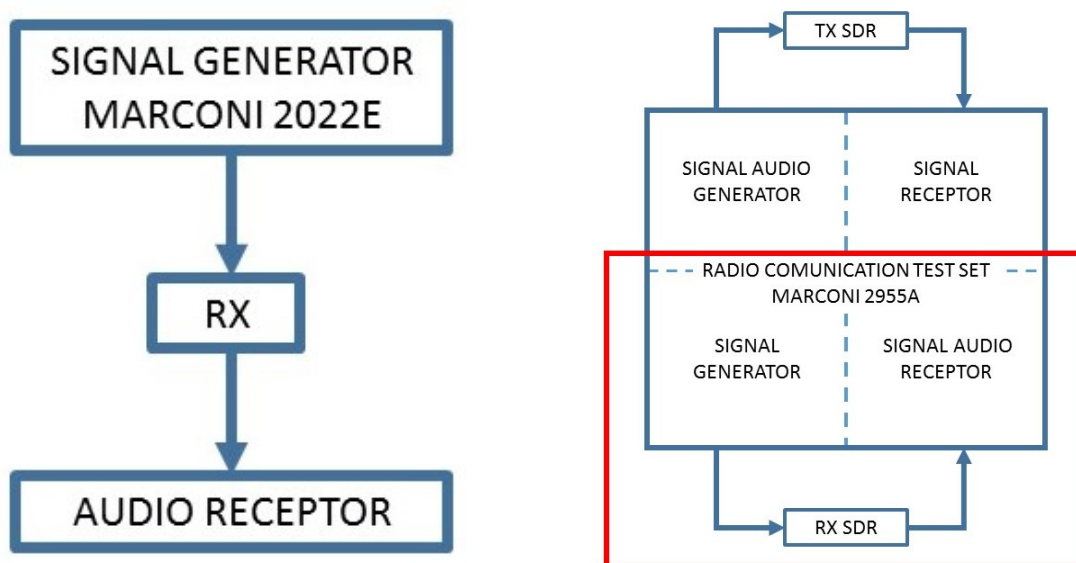
Figure 3.8 – HP 8903B audio analyzer

However, the use of these two devices has the problem that is more difficult to obtain a ratio between the signal emitted and the signal received. An alternative to use the method described before is using a radio communication test set. In the laboratory we have available the Marconi 2955 communication test set (3.9). This device has a signal generator and receptor as well as an audio generator and receptor. In other words, we have two devices in one.



Figure 3.9 – Marconi 2955 communication test set

Figure 3.10 shows a schematic for each one of the measurement alternative described before. Figure 3.10a shows the schematic of the method that uses a signal generator and an audio receptor while figure 3.10b shows the schematic of the method that uses a communication test set. We have to say that radio communication test set also allows to measure the sensibility of a SDR transmitter. However, in our case we only have to measure the sensibility of a SDR receptor (section inside the red box).



(a) Using a signal generator and a audio receptor

(b) Using a communication test set

Figure 3.10 – Sensibility measurement schematics

Using a radio communication test set has the advantage that we can measure the signal ratio easily between the output signal and the input signal, which allows us to obtain the



**SINAD** ratio. Moreover, using only one device instead of two simplifies all the process.

In the beginning of this section, we said that we have to find the minimum energy level for the signal that allows us to receive the signal correctly in the audio receptor but, what means receive the signal correctly in engineering terms? If we observe the **FUNCube Dongle** technical specifications (3.2) we can see that we need  $0.15 \mu\text{V}$  to obtain 12 dB **SINAD** at 145 MHz and 435 MHz. We can deduce that 12 dB means a good quality in the signal reception so, in order to measure the **SDR** sensibility we will find the minimum energy level for the signal that allows us to receive a signal in the audio receptor that gives us 12 dB **SINAD**.

## CHAPTER

# 4

# TESTBED ANALYSIS

Before programming the [Testbed](#), we have to understand how it is designed and implemented. Moreover, the [Testbed](#) designed and built by Victor Burgos was made one year before starting the present project so, it is logical to think that some parts of the [Testbed](#) can be damaged or work wrong. In this chapter we are going to analyze each part of the [Testbed](#) (Air-bearing, air compressor and [PCB](#)) and solve the problems that we have found. Additionally, the microprocessor has pins without use so, at the end of the chapter (section [4.3](#)) we are going to add a buzzer and a button to the [PCB](#). Finally we will explain what is a [PID](#) controller for a better understanding of its implementation in section [4.4](#)

## 4.1 Air-bearing and air compressor analysis

### 4.1.1 Air-bearing analysis

An important element of the [Testbed](#) is the [air-bearing](#). Victor Burgos built two (figures [4.1](#) and [4.2](#), both of them with a 3D printer.



**Figure 4.1** – *Pink air-bearing*



**Figure 4.2** – *Green air-bearing*

However, they had not the desired performance. The first one (pink [air-bearing](#)) had a rough surface after the printing and, consequently, we can not achieve a friction-less scenary when we eject air under pressure.

On the other hand, the green [air-bearing](#) had a smooth surface (although it was not smooth enough as we will explain later) but it had manufacturing defects becuase it had little holes in their base and, when we ejected air under pressure, some of this air escaped from the holes. Consequently, the airflow was not enogh to achieve a friction-less scenary. To solve this problem we have to repair the [air-bearing](#) covering the holes. We will dissolve plastic for 3D printing in acetone to create a paste. We will cover the [Testbed](#) base with this paste and when it will be dry, the holes will be covered. [Figure 4.3](#) shows the acetone and the plastic used to create the paste.



Figure 4.3 – *Air-bearing repair*

4

Figure 4.4 shows the *air-bearing* base before (4.4a) and after (4.4b) the reparation. We can observe how the holes have been covered.



(a) *Before*

(b) *After*

Figure 4.4 – *Air-bearing before and after the reparation*

Once the *air-bearing* was repaired, the air did not scape from the base and we could achieve a friction-less scenary in the green *air-bearing*. However, although the surface was smooth, it was not smooth enough so we needed air under very much pressure to obtain a friction-less scenary. This was not optimum because higher the pressure is, quicker the

compressor tank gets empty. One of the problem that Victor Burgos had in his test was that the compressor tank gets empty very quickly and he did not have enough time for his tests.

The solution to this problem was to obtain a new [air-bearing](#) made by another material and build by another method because it was demonstrated that using plastic from 3D printing we can not get a smooth enough surface. The tutor of this project, Andres Roldan Aranda, gets a new [air-bearing](#) with a surface made with epoxy resin with calcium carbonate and glass fiber, using a PVC tube as base. Figure 4.5 shows the new [air-bearing](#).



**Figure 4.5** – *Air-bearing* made by epoxy resin

The epoxy resin allows us to obtain a smoother surface than using plastic and, consequently, we can achieve a friction-less scenario with air under relatively low pressure.

#### 4.1.2 Air compressor analysis

The [air-bearing](#) needs air under pressure to create the friction-less scenario so it is necessary an air under pressure source. Victor Burgos used the air compressor that was available in the laboratory, the CP2525 model, of Black&Decker (figure 4.6). However, this air compressor has a little air tank, so the experiments test can not last a long time. Table 4.1 shows the characteristics of the air compressor.



**Figure 4.6** – *CP2525 air compressor*[20]

Oil lubricated
24 liters tank
Pressure regulator and 2 pressure gauges
Plastic shroud protects hot and rotting parts
230 V / 50 Hz
Maximum working pressure of 8 BAR and the motor has a power of 2HP peak

**Table 4.1** – *CP2525 air compressor characteristics*

The time that the friction-less scenario can be kept depends on some elements apart from the air tank capacity. It is logical to think that while heavier is the platform [Testbed](#), it is necessary air under more pressure to create the friction-less scenario. Consequently, while higher is the air-pressure, the air tank gets empty earlier. If we want to create the friction-less scenario with this air compressor, the air pressure and the weight must be minimum.

In order to know exactly how much time the friction-less scenario last according with the weight and the air pressure, the friction-less time has been measured for weights from 0.75 Kg to 2 Kg and for air pressure from 1 kg/cm<sup>2</sup> to 4 kg/cm<sup>2</sup>. It has been considered that there is friction-less scenario when the [Testbed](#) platform turns without applying any external force. Results can be observed in table 4.2. Figure 4.7 shows this results graphically.

Pressure	0.75 kg	1 Kg	1.25 Kg	1.5 Kg	1.75 Kg	2 Kg
4	01:33,64	01:05,83	00:52,28	00:48,53	00:46,09	00:40,92
3	01:39,87	01:10,90	00:55,38	00:50,80	00:53,39	00:47,38
2	01:46,86	01:18,05	01:11,82	00:57,80	00:55,70	00:49,70
1	02:40,00	01:39,75	01:22,21	01:17,92	01:05,77	01:01,46

Table 4.2 – Variation of the friction-less scenary temp with the weight and the air pressure

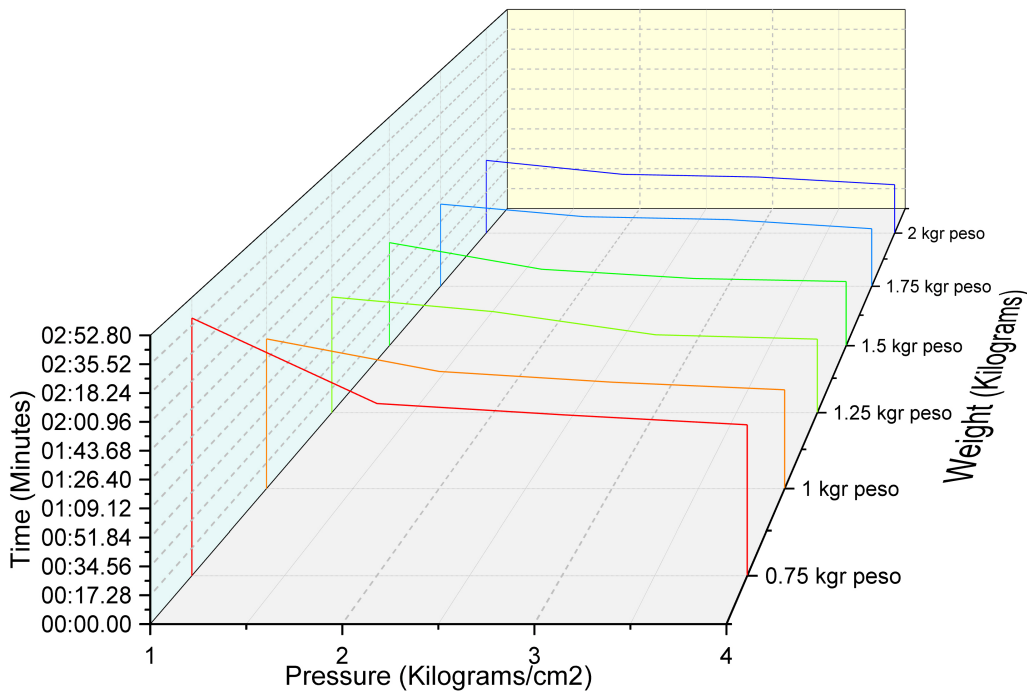


Figure 4.7 – Graph of the variation of the friction-less scenary time with the weight and the air pressure

As we said in section 1.2 ([52]), an 1U Cubesat cannot weigh more than 1,33 Kg. If we focus in 1.25 Kg colum (table 4.3), friction-less time is between 01:22,21 minutes (1 kg/cm<sup>2</sup>) and 00:52,28 minutes (4 kg/cm<sup>2</sup>). We have to take into account that if we use air under low pressure (for example, 1 kg/cm<sup>2</sup>), the actuators will need more power to move the Testbed because the friction between the air-bearing and the Testbed platform will be higher than using air under high pressure (4 kg/cm<sup>2</sup>). If we use 3 kg/cm<sup>2</sup> of air pressure, we have around one minute of friction-less time. This time is not enough because the stabilization of the Testbed tend to be very slow (it could last more than a minute). Moreover, during the programing of the Testbed we will have to do a lot of test and it will be a hard task if we have to wait to fill the air tank each minute of testing. For these reasons it would be desirable to adquire a new air pressure source that allows us to work during a long time.

Pressure	1.25 Kg
4	00:52,28
3	00:55,38
2	01:11,82
1	01:22,21

**Table 4.3** – Variation of the friction-less scenary time with the air pressure for 1,25 kg

Andrés Roldan, the tutor of this project, gets a new air compressor from a disuse air-conditioning system. This new air compressor, which can be observed in picture 4.8, works in a different way to the CP2525 air compressor because it does not need to store the air in a tank before ejecting it under pressure and, in theory, we have air flow under pressure without time limit. In practice, the air compressor gets hot when has been working during a long time and we have to switch it off until it will be cold again. With this new air compressor we have solved the friction-less time problem.



**Figure 4.8** – New compressor used for the *air-bearing*



## 4.2 PCB analysis

In this section we are going to see how the **PCB** of the **Testbed** is built and to analyze the sensors and actuators. The **PCB** has the objective of connect the sensors, actuators and the **LCD** screen with the microprocessor and the power supply. The **PCB** was designed by Victor Burgos using **Altium Designer**. The [following page](#) shows the schematic of the **PCB**, with every subsystem included on it.

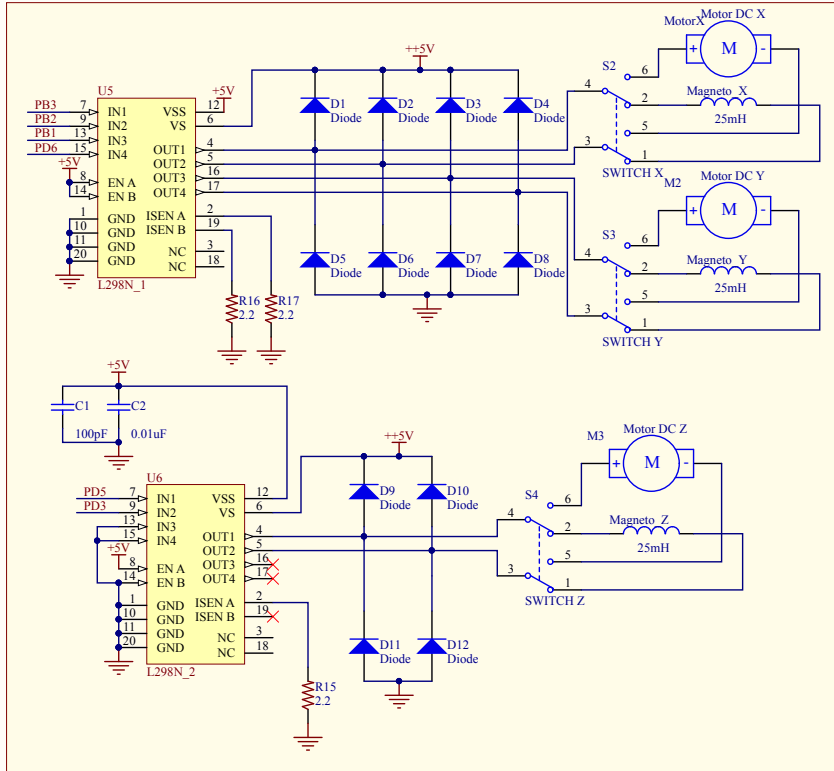
As it can be seen in the schematics, the main parts of the **PCB** are:

- Microprocessor ATMEGA328P
- Display **LCD**
- Power supply
- LSM303 and MPU6050 sensor devices and light sensors
- Driver motors (actuators)
- Xbee module wireless communication

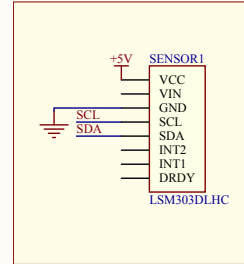
# 4

Each one of these elements will be described in one subsection of this chapter (sections from [4.2.1](#) to [4.2.6](#)).

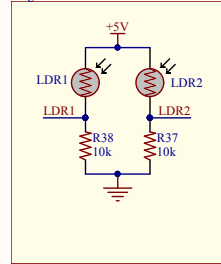
DRIVER MOTOR L298P



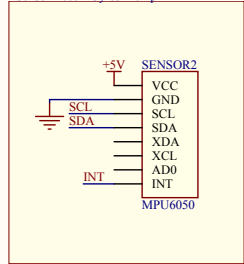
Sensor Accel+Magnetom+Temp



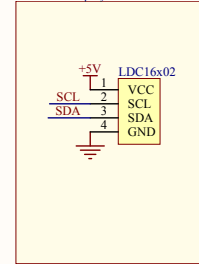
Light Sensor



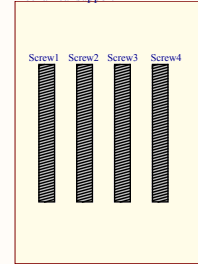
Sensor Accel+Gyros+Temp



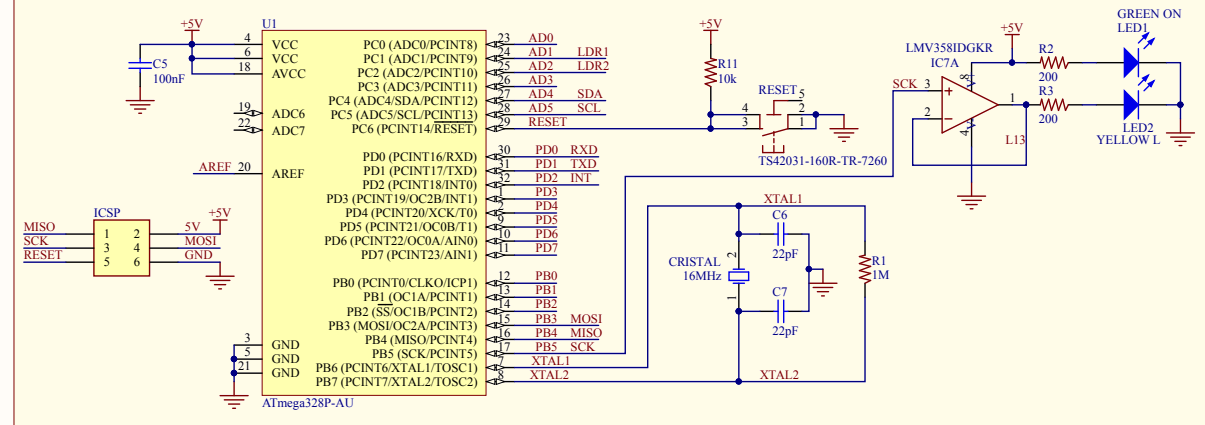
HD44780 Display



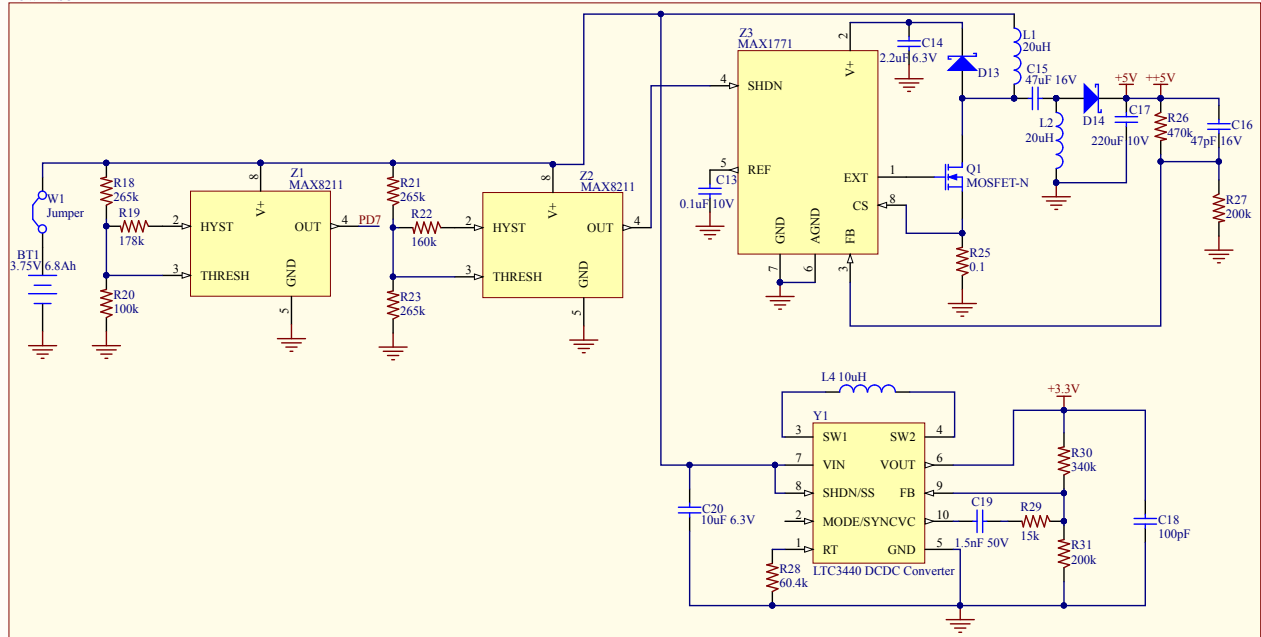
Mechanical support



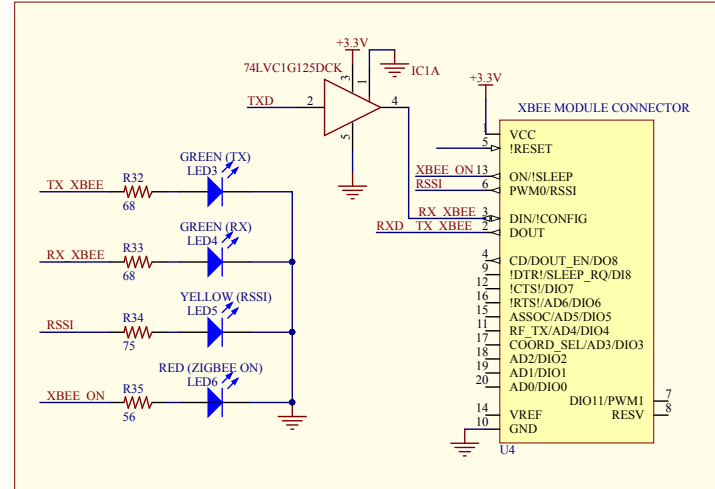
ATMEGA328P-AU



POWER SUPPLY



XBEE MODULE CONNECTOR + SWITCH + LEDs



Designer's signature	Sheet title: <b>Testbed PCB Schematic</b>	Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n. 18001 Granada, Granada, Spain Sr. Andrés Beldán Aranda
Supervisor's signature	Project title: <b>Testbed for Cubesat GranaSAT</b>	
	Designer: <b>Victor Burgos González</b>	
	Date: <b>23/06/2015</b> Revision: <b>V10</b>	Sheet * of *



To understand the connections between the different elements inside the PCB is very important for the programming because we have to know where the sensors and the actuators are connected, as well to know if the microprocessor have some pins without any connection in order to connect new sensors or actuators (as we will do in the future).

### 4.2.1 Microprocessor analysis

The microprocessor used in the Testbed is the ATmega328P model in SMD model to save space and weight for the PCB. The main features of the ATmega328P can be seen in table 4.4. The pin-out of the microprocessor can be seen in figure 4.9.

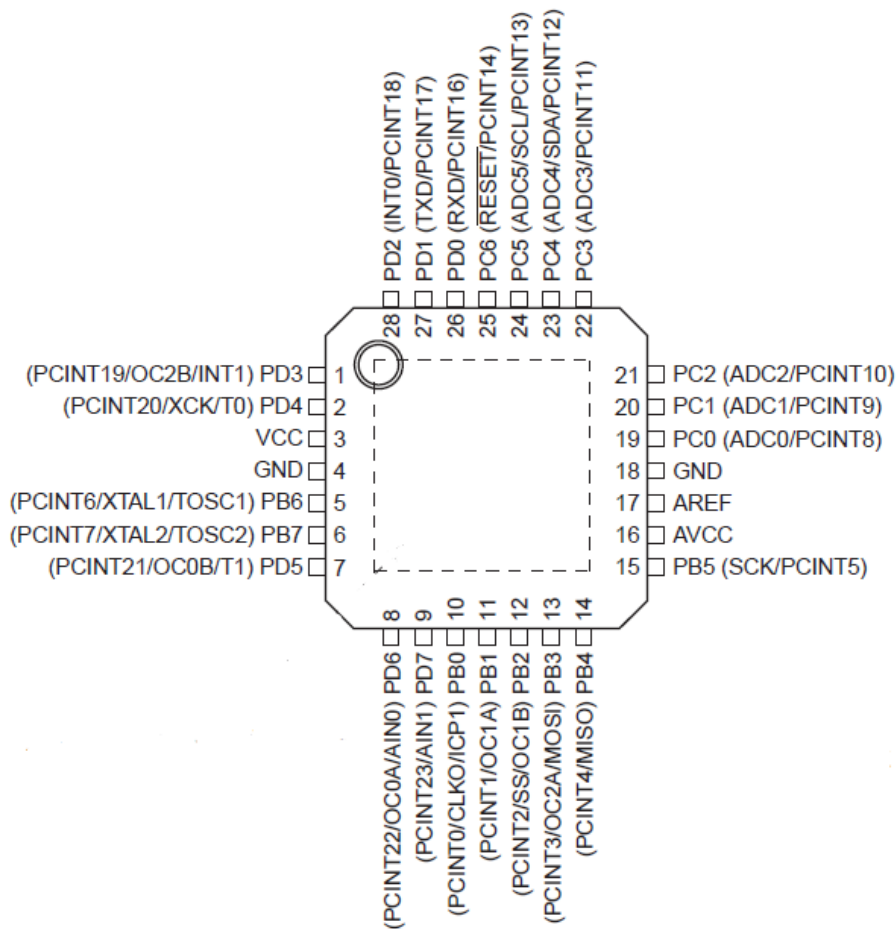
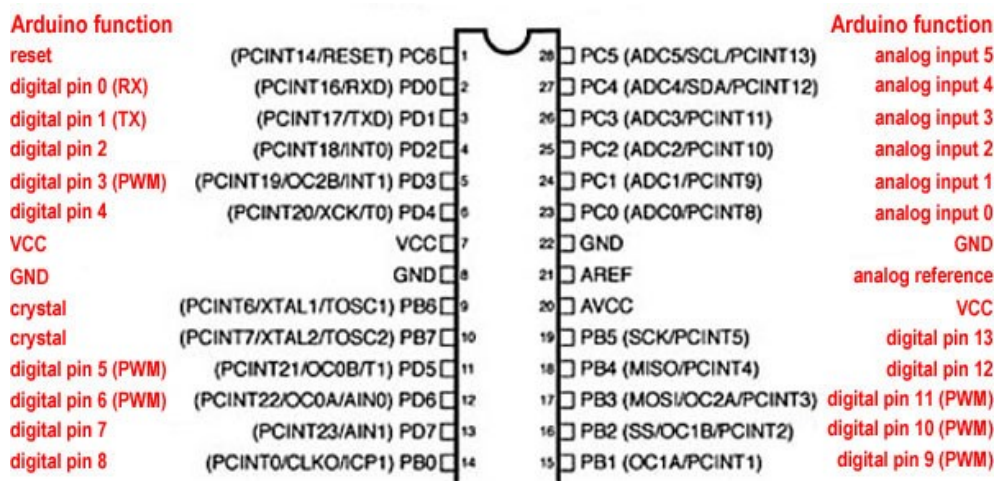


Figure 4.9 – ATmega328p pin-out [19]

Parameter	Value
Flash (kBytes)	32 kBytes
Pin Count	32
Max. Operating Freq. (MHz)	20 MHz
CPU	8-bit AVR
# of Touch Channels	16
Max I/O Pins	23
SPI	2
ADC Channels	8
ADC Resolution (bits)	10
ADC Speed (ksps)	15

**Table 4.4** – *ATmega328 main features [40]*

The microprocessor is accompanied by a **ICSP** port, which is used for program it with a external programmer. It is thought to use the microprocesor as an **Arduino UNO**, so it is installed an external 16 MHz clock in the **PCB**, as well as a button to reset the **Testbed**. Moreover, if we want to use the microprocessor as an **Arduino**, we have to install in it an **Arduino bootloader**. This process is described in appendix **A**. However, **Arduino** boards use different names to the Atmega328 for the pin-out. In figure **4.10** we can see the correspondece between the ATmega328 microcontroller (in black words) and the **Arduino** boards (in red words).



**Figure 4.10** – *Pin-out for ATmega328p and Arduino boards [1]*

If we want to use the microprocessor as an **Arduino**, apart from install a 16 MHz clock, we have to install a **bootloader**. A **bootloader** is a piece of firmware in the microcontroller that allows installing new firmware without using an external programmer [2]. Moreover,

an [Arduino bootloader](#) is needed if we want to program the microprocessor from [Arduino IDE](#). In appendix [A](#) it will be explained how to install a [bootloader](#) in the ATmega328p microcontroller. However, we have to take into account that the watchdog do not work with all [bootloader](#) so we will have to choose one where the watchdog works.

### 4.2.2 Display LCD

The [PCB](#) has a 16×2 [LCD](#) display and it is used to seeing data from the sensors and from the ground station in real time. Moreover it will be used to facilitate the visualization of the data without the wireless connection. The [LCD](#) has a I<sup>2</sup>C converter incorporated on it, because of the insufficient number of pins that the microprocesor has and its easy functionality [[21](#)](figure [4.11](#)).



**Figure 4.11** – LCD display and I<sup>2</sup>C converter (red board)[[21](#)]

### 4.2.3 Power system analysis

The desing and implementation of the [PCB](#), which includes all the electronics components of the [Testbed](#), was made by Victor Burgos Gonzalez in his bachelor thesis [[21](#)]. After the testing, all the components worked correctly except the power supply. For this reason, it was necessary to connect the [PCB](#) directly to a fixed power suply instead of using our battery and, consequently, losing mobility.

The problem was that it initially was designed as a step down. However, our battery has a nominal voltage of 3,75 volts and we need 5 volts to power the microprocessor (In other words, we need a step up power supply instead of a step down) so it is necessary to replace the original power supply by a new one. The new power supply will be composed by four main components:

- **Battery.** Battery that allows power the system whitout using a fixed power supply.

- **Step up.** Module that increases the voltage from 3,75 V (battery voltage) to 5 V
- **Charger module.** This module allows to charge the battery directly from the PCB, which it was not included in the original design.
- **Voltage regulator.** We need a 3,3 V regulator to power the XBee wireless module.

Knowing this, we have two different alternatives to get this components, buy it or build it by ourselves. In the case of the battery, we will use the same battery that Victor Burgos used in his bachelor thesis ([21]). Moreover, there are voltage regulators available in the laboratory, so we have not to buy a new one.

On the other hand, table 4.5 shows the prices of the step up and charger module available in [es.aliexpress.com](https://www.aliexpress.com). Built this components by ourselves is more expensive because we would have to buy all the components of the modules (resistors, leds, inductors, integrated circuits...), print the circuit in copper and spend time in the desing process.

Step Up	0,39 €	[17]
Charger module	0,91 €	[16]

**Table 4.5** – Prices for step up and charger module

However, in section 1.1 we said that one of the objectives of doing this project was to learn about task in which I had little knowledge. For this reason we decided to design the step up module in order to learn how to design and build PCBs although at the end of the process we bought the step up instead of implementing it because the cost must be minimum. This process will be described in chapter 6.

#### 4.2.4 Sensors analysis

The Testbed has several sensors that allow to compile information and measurements from the environments in order to the ADCS can determine the attitude. These sensors are:

- Gyroscope
- Accelerometer
- Magnetometer
- Light sensors

In this section it will be described the different elements that contain these sensors as well as the way to use them to obtain measurements correctly.

#### 4.2.4.1 MPU6050

The MPU-6050 (figure 4.12) is an integrated device that combines a 3-axis MEMS gyroscope, a MEMS 3-axis accelerometer and a DMP in a small 4x4x0.9 mm packaged. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore it captures the x, y, and z channel at the same time. The sensor uses the I<sup>2</sup>C bus to interface with the microprocessor [45]. Table 4.6 shows the general features of the devices, while in tables 4.7 and 4.8 it can be seen the specific features of the gyroscope and accelerometer respectively.



Figure 4.12 – MPU-6050 [45]

9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
Auxiliary master I <sup>2</sup> C bus for reading data from external sensors (e.g. magnetometer)
3.9 mA operating current when all 6 motion sensign axes and the DMP are enabled
VDD supply voltage range of 2.375 V-3.46 V
Flexible VLOGIC reference voltage supports multiple I <sup>2</sup> C interface voltages
Smallest and thinnest QFN package for portable devices: 4X4X0.9 mm
Minimal corss-axis sensitivity between the accelerometer and gyroscope axes
1024 byte FIFO buffer reduces powe consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
Digital-output temperature sensor
User-programmable digital filters for gyroscope, accelerometer and temp sensor
10.000 g shock tolerant
400 kHz Fast Mode I <sup>2</sup> C for communicating with all registers

**Table 4.6** – MPU6050 general features [40]

Digital-output X-, Y- and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of $\pm 250$ , $\pm 500$ , $\pm 1000$ and $\pm 2000^\circ/\text{sec}$
External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
Integrated 16-bit ADCs enable simultaneous sampling gyros
Enhaced bias and sensitivity temperature stability reduces the need for user calibration
Improved low-frequency noise performance
Digitally-programmable low-pass filter
Gyroscope operating current: 3,6 mA
Standby current: $5\mu\text{A}$
Factory calibrated sensitivity scale factor
User self-test

**Table 4.7** – MPU6050 gyroscope features [40]



Digital-output triple-axis accelerometer with a programmable full scale range of $\pm 2g$ , $\pm 4g$ , $\pm 8g$ and $\pm 16g$
Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
Accelerometer normal operating current: $500\mu A$
Low power accelerometer mode current $10\mu A$ at 1.25Hz, $20\mu A$ at 5 Hz, $60\mu A$ at 20 Hz, $110\mu A$ at 40 Hz
Orientation detection and signaling
Tap detection
User-programmable interrupts
High-G interrupt
User self-test

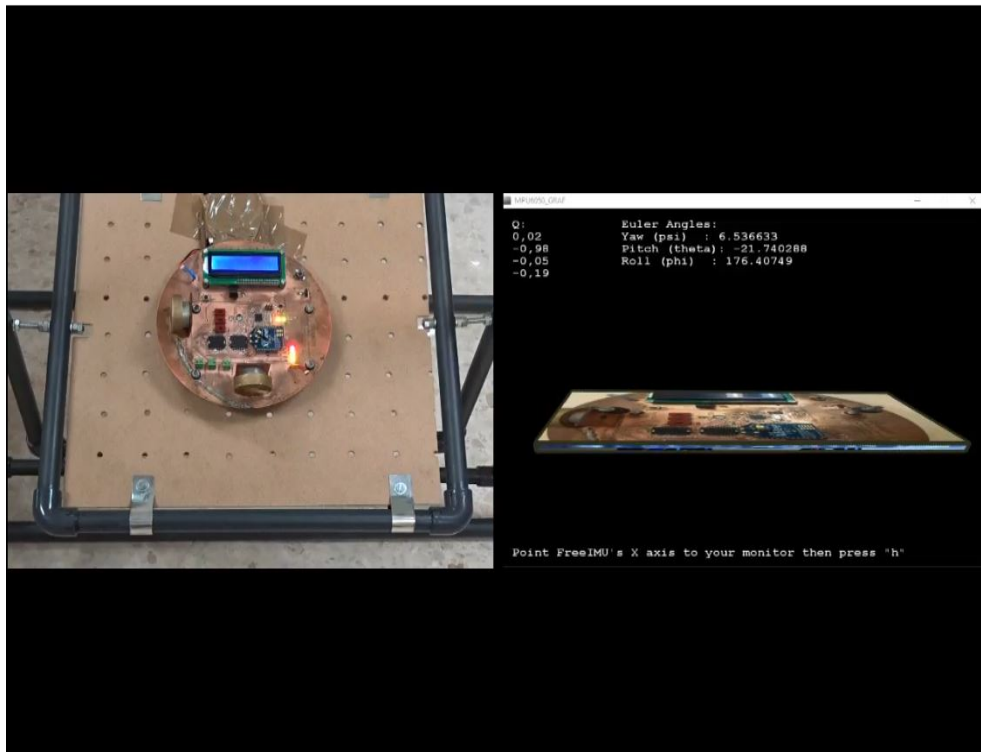
**Table 4.8** – MPU6050 accelerometer features [40]

4 Reading the raw values for the accelerometer and gyroscope is a easy task. The sleep mode has to be disabled, and then, the registers for the accelerometer and gyroscope can be read. Moreover, the sensor also contains a 1024 byte FIFO buffer. The sensor values can be programmed to be placed in the FIFO buffer and it can be read by the microprocessor. However, we have to remember that these are raw data, so it does not give us real information, we have to process the data before. The sensor has a DMP, also called “Digital Motion Processing Unit”. This DMP can be programmed with firmware and is able to do complex calculations with the sensors values [45].

The DMP can do fast calculations directly on the chip. This reduces the load for the microcontroller. The DMP is even able to do calculations with the sensor values of another chip, for example, a magnetometer connected to the I<sup>2</sup>C bus [45].

Jeff Rowberg, a software engineer, has done an excellent job programming a library for MPU6050. This library, included in *I<sup>2</sup>Cdevlib* library, allow us to use the DMP unit for calculating real measurements from raw data in an easy way. We can find more information about this library in [34] and it can be downloaded from [31].

As example, using the MPU6050 library and Processing 2 software ([46]) we can observe in the following video how the figure in the computer screen (right part) turn in the same way that we turn the PCB of the Testbed (left part). The code used can be downloaded from [44].



Video 4.1 – Gyroscope use example

4

If there is any problem to see the video correctly, see [https://www.youtube.com/watch?v=k8KPIu5F3T4&feature=youtu.be&ab\\_channel=GranaSATTeam](https://www.youtube.com/watch?v=k8KPIu5F3T4&feature=youtu.be&ab_channel=GranaSATTeam).

With the MPU6050 library we can obtain the yaw, pitch and roll angles. These angles, very used in navigation and aerospace, describe the direction (yaw), elevation (pitch) and rotation angle (roll) of the aircraft body. Figure 4.13 shows this reference system graphically.

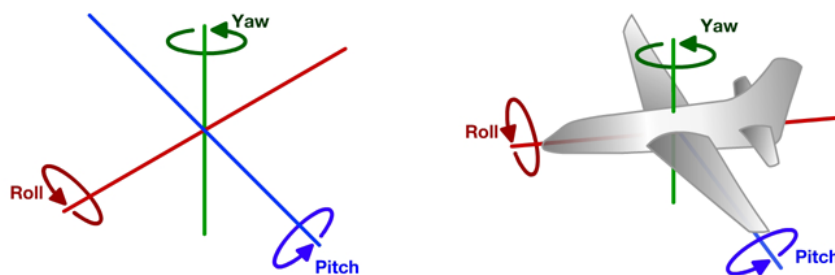


Figure 4.13 – Yaw, pitch and roll reference system [28]

In our case, we will consider yaw angle as the rotation around z angle, pitch as the rotation around y angle and roll as x angle as it is shown in figure 4.14. The x, y and z axis considered for the PCB are shown in figure 4.15.

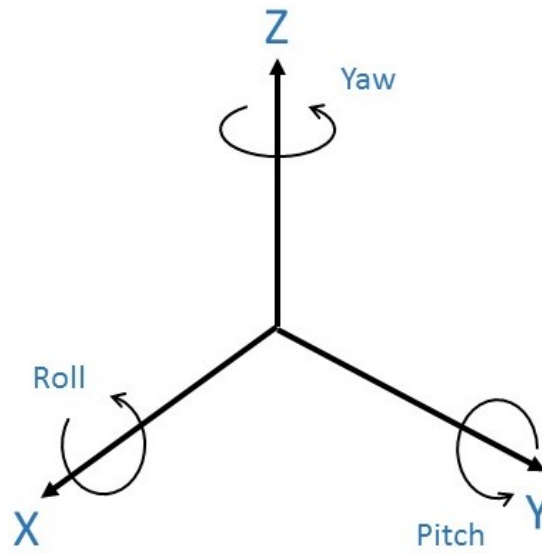


Figure 4.14 – Axes used as system reference [28]

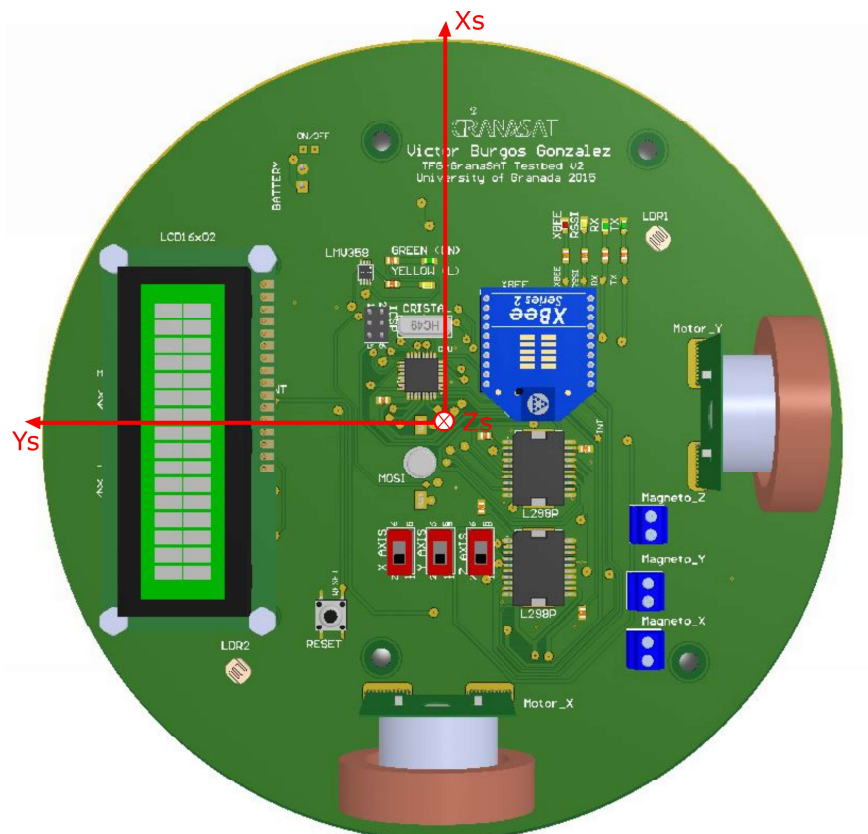


Figure 4.15 – PCB reference system [21]

Once we have the yaw, pitch and roll angles, using equations 4.2.1, 4.2.2 and 4.2.3 respectively we can express the rotating of each axis in degrees from -180 to 180.

$$GyroZ = YawAngle \times \frac{180}{\pi} \quad (4.2.1)$$

$$GyroY = PitchAngle \times \frac{180}{\pi} \quad (4.2.2)$$

$$GyroX = RollAngle \times \frac{180}{\pi} \quad (4.2.3)$$

#### 4.2.4.1.1 MPU6050 calibration

To be ensure that we are using the [DMP](#) correctly, we have to calculate the calibration offsets. These offsets are using to calibrate the MPU6050's internal [DMP](#), but can be also useful for reading sensors. To calculate these offset we will run an [Arduino](#) sketch <sup>1</sup>, which can be downloaded from [\[5\]](#). The results obtained in the sketch output are:

```
Sensor readings with offsets: 6 3 16377 0 -2 0
Your offsets: -1323 1262 5425 43 -30 32
Data is printed as: accelX acely accelZ giroX giroY giroZ
```

Knowing the offset values, we can calibrate the MPU6050's internal [DMP](#) during the software programing with the next sentences:

```
mpu.setXAccelOffset(-1323);
mpu.setYAccelOffset(1262);
mpu.setZAccelOffset(5425);
mpu.setXGyroOffset(43);
mpu.setYGyroOffset(-30);
mpu.setZGyroOffset(32);
```

#### 4.2.4.2 LSM303

The LSM303 (figure 4.16) is a system-in-package featuring a 3D digital acceleration sensor and a 3D digital magnetic sensor. Includes an I<sup>2</sup>C serial bus interface that supports standard and fast mode 100 kHz and 400 kHz. The system can be configured to generate interrupt signals by inertial wake-up/free-fall events as well as by the position of the device itself. Thresholds and timing of interrpt generators are programmable by the end user. Magnetic and accelerometer blocks can be enabled or put into power-down mode separately to save

<sup>1</sup>The MPU6050 must be in horizontal position

power. In table 4.9 are enumerated the LSM303 features. Image 4.17 shows the reference system of the device.



Figure 4.16 – LSM303 [29]

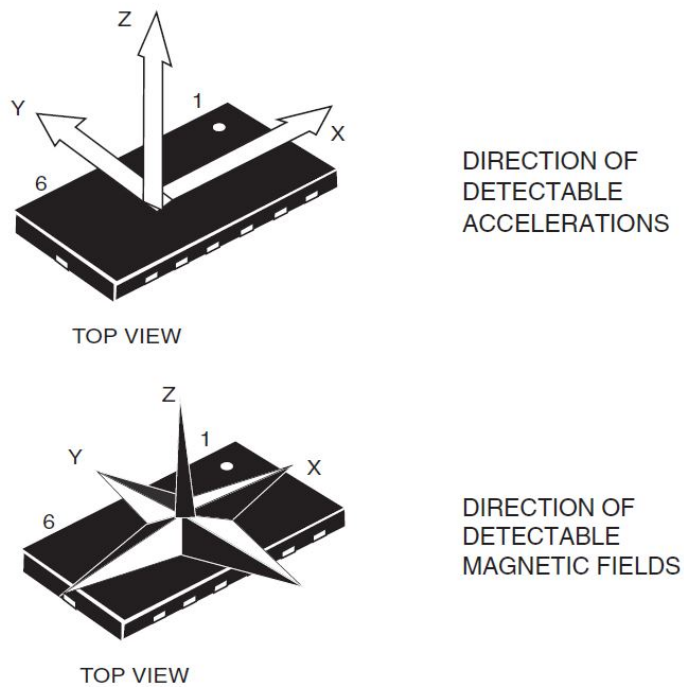


Figure 4.17 – LSM303 reference system [49]

3 magnetic field channels and 3 acceleration channels
From $\pm 1.3$ to $\pm 8.1$ gauss magnetic field full scale
$\pm 2g/\pm 4g/\pm 8g/\pm 16g$ linear acceleration full scale
16-bit data output
I <sup>2</sup> C interface
Analog supply voltage 2.16 V to 3.6 V
Power-down mode / low-power mode
2 independent programmable interrupt generators for free-fall and motion detection
Embedded FIFO
6D/4D-orientation detection
ECOPACK@RoHS and “Green” compliant

**Table 4.9** – *LSM303 features [49]*

In the same way that MPU6050, there is a library for LSM303 (included in *I<sup>2</sup>Cdevlib* library too) that allows us to use the magnetometer and accelerometer easily. However, the LSM303 device has not a module similar to the **DMP** in the MPU6050 and, consequently, we obtain the raw data without processing. To obtain the real data of the measurements, we have to multiply the raw data by a scaling factor to obtain the accelerometer data in mg and the magnetometer data in mG. The scaling factor depends on the scale used with the accelerometer and the magnetometer and it can be seen in tables 4.10 and 4.11 respectively.

Scale	Scaling factor
$\pm 2$	1
$\pm 4$	2
$\pm 8$	4
$\pm 16$	8

**Table 4.10** – *Scaling factor for LSM303 accelerometer [49]*

Scale	Scaling factor (X and Y axis)	Scaling factor (Z axis)
$\pm 1.3$	1/1100	1/980
$\pm 1.9$	1/855	1/760
$\pm 2.5$	1/670	1/600
$\pm 4.0$	1/450	1/400
$\pm 4.7$	1/400	1/355
$\pm 5.6$	1/330	1/295
$\pm 8.1$	1/230	1/205

**Table 4.11** – *Scaling factor for LSM303 magnetometer [49]*

In order to understand how these scale factors have been obtained, we have to calculate the real data for the accelerometer without using it. The LSM303 device has 16 bit data output. This means that the device returns values between -32768 and 32767. The corresponding acceleration to these data depends of the scale. In our case, we will use a  $\pm$  scale because it is the scale by default and we will not need to measure higher values. If the scale is  $\pm 2$ , this means that -32768 corresponds to an acceleration of -2g and 32768 corresponds to 2g. Knowing that, it is easy to calculate the intermediate values. The calculation is expressed in the equations from 4.2.4 to 4.2.6 for each axis.

$$AccX = RawX \times \frac{9.81}{16384.0} \quad (4.2.4)$$

$$AccY = RawY \times \frac{9.81}{16384.0} \quad (4.2.5)$$

$$AccZ = RawZ \times \frac{9.81}{16384.0} \quad (4.2.6)$$

## 4

In our case we will use a  $\pm 1.3$  scale for the magnetometer because, in the same way that the accelerometer, is the scale by default and we will not need to measure higher values. With the scaling factor we can obtain the real data in milligauss, but we have to convert the values in Tesla <sup>2</sup> units because it is the international system unit. However, 1 Tesla is a big amount of density magnetic field, so we will express the data in  $\mu$ T. So we will have to multiply the raw values by  $\frac{1}{1100}$  (X and Y axis) or by  $\frac{1}{980}$  (Z axis) to obtain mG and then divide it by 10 to obtain  $\mu$ T. This calculation is expressed in the equations from 4.2.7 to 4.2.9 for each axis.

$$MagX = RawX \times \frac{1}{1100} \times \frac{1}{10} \quad (4.2.7)$$

$$MagY = RawY \times \frac{1}{1100} \times \frac{1}{10} \quad (4.2.8)$$

$$MagZ = RawZ \times \frac{1}{1100} \times \frac{1}{10} \quad (4.2.9)$$

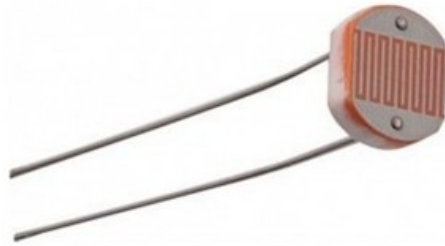
#### 4.2.4.3 Light sensors

We will use LDRs as light sensors (figure 4.18). The LDR are light sensitive devices that, in the dark, their resistance is very high, sometimes up to 1M $\Omega$ . When the LDR is exposed to light, the resistance drops dramatically, even down to a few ohms [33]. The higher the

---

<sup>2</sup>1 Tesla is equal to 10000 Gauss

light intensity is, the lower is the resistance.



**Figure 4.18** – *Light dependent resistor [33]*

In the [PCB](#), we have two [LDRs](#). To differentiate between them, one has their terminals covered by red heat-shrink wire and the other one covered by blue heat-shrink wire. The [LDRs](#) are connected to analogical pins of the microprocessor. When we read from these pins, we obtain values between 0 and 1023. The higher the light intensity is, the higher is the read value.

In order to have a greater control of the reading data, we have measured the maximum and minimum value that the light sensors return. First, we have read the light sensors with a direct exposure to a light source (in our case, we have used the lantern mobile phone) obtaining 413 for red [LDR](#) and 437 for blue [LDR](#). Then, we have covered the [LDRs](#) and we have measured their values in the darkness, obtaining 0 for both [LDRs](#). [Table 4.12](#) summarizes these measurements.

	Maximum value	Minimum value
<b>Red light sensor</b>	413	0
<b>Blue light sensor</b>	437	0

**Table 4.12** – *Minimum and maximum values for light sensors*

During the programming, the reading values will be mapping between 0 and 500 to have a better understanding of the light detected.

#### 4.2.5 Actuators analysis

In [section 1.2](#) it was said that one of the most common actuators used in systems was the reaction wheels. In the [Testbed](#) it was included three reaction wheels, one for each axis that make the [Testbed](#). These wheels are able to turn thanks to they are fixed to electric motors. Victor Burgos, in his bachelor thesis [\[21\]](#) made a complete analysis of the performance of each one of the motors. However, this analysis was made without any load or, in other words, without the reaction wheels fixed. The motors had little differences in their performance and



it is logical to think that these differences will be higher when the reaction wheels will be fixed. In this section we are going to analyze the reaction wheels of each axis measuring the **RPM** that reach the wheels according to the voltage between their terminals. This analysis will be very useful for programming the **PCB** because the angular moment of the reaction wheels depends directly of their **RPM** and the higher the **RPM** are, the higher the inertial torque is.

The **RPM** will be measured with a photo tachometer DT-2234B from Lutron Electronics (figure 4.19). In table 4.13 we can see its technical specifications.

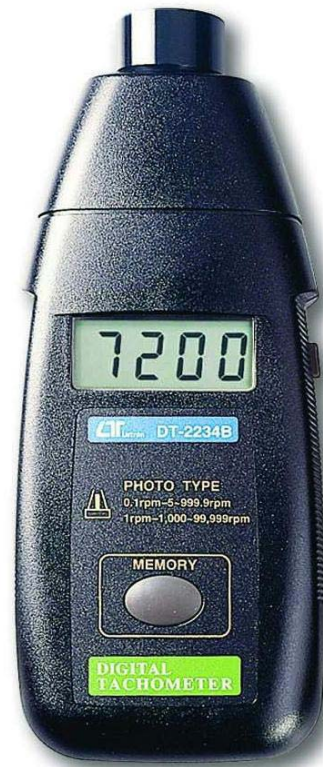


Figure 4.19 – Photo tachometer [43]

Range (RPM)	5 to 99999
Resolution for <1000 RPM	0.1
Resolution for >1000 RPM	1
Accuracy	$\pm (0.05\% + 1 \text{ digit})$
Sampling time (s)	1
Operating temperature (C)	0 to 50
Charge termination current (mA)	150
Discharge temperature range (C)	-10 to +60

**Table 4.13** – *DT-2234B photo tachometer specifications [43]*

The motors are connected to analogical pins of the microcontroller. That allows us to power the motors with PWM pulses. The microcontroller can write 8 bits data output so, we can write value between 0 and 255. The higher the value is, the higher the voltage is in the motor terminals and, consequently, the higher the RPM are. Moreover, the motors have H bridge and we can change between forward and backward direction easily.

Although in theory the voltage in terminals should be equal for the same PWM pulse whatever the direction was, in practice this is not true because the path in the PCB is different for forward and backward direction. For this reason, the RPM of the motors change with the direction.

We have to take into account that for low PWM values the motors would not turn the wheels because the voltage between their terminals is not enough. We have measured the minimum PWM value that is needed to start to turn each one of the three wheels. These values are shown in table 4.14.

Wheel	Forward value	Backward value
X axis wheel	95-105	95-105
Y axis wheel	65-75	50-55
Z axis wheel	60-65	35-45

**Table 4.14** – *Minimum PWM value necessary for each axis wheel to start to turn*

Although the wheels start to turn for the PWM values said before, it is possible to measure how many RPM the wheels reach if we start with a higher value and then we reduce it. The inertia will help the wheels to turn.

The method of measurement of the RPM is not very accurate. For this reason, we will take three measurements and we will take the average as final result. Moreover, we have to take into account that the Testbed should be in the same conditions for each measurement. For example, if the Testbed has a small inclination, the gravity can help the wheel to turn and the RPM are higher.

#### 4.2.5.1 X axis motor

The X axis motor is responsible for moving the [Testbed](#) in roll angle. In [table 4.15](#) we can see the correspondence between the [PWM](#) pulse value in the analgical pins of the microcontroller and the voltage measured in the motor terminal for forward and backward direction.

PWM Pulse	Voltage forward (V)	Voltage Backward (V)
0	0	0
15	0.04	0.03
30	0.12	0.11
45	0.45	0.47
60	1.18	0.85
75	1.28	1.17
90	1.47	1.46
105	1.68	1.69
120	1.85	1.92
135	2.06	2.1
150	2.23	2.25
165	2.38	2.43
180	2.51	2.58
195	2.66	2.71
210	2.84	2.84
225	2.98	3.04
240	3.15	3.16
255	3.42	3.43

**Table 4.15** – Correspondence between voltage and [PWM](#) pulse in X axis motor

In [tables 4.16](#) and [4.17](#) we can see each one of the three measurements of the [RPM](#) according with the [PWM](#) pulse for forward and backward direction respectively. We have taken the [PWM](#) pulse each 15 units, in order to shorten the measurement process.

In [figure 4.20](#) we can see a graphical representation of the [RPM](#) average with the [PWM](#) pulse for forward and backward direction, as well as a comparison between them. We can see how, in general terms, the [RPM](#) are higher for forward than for backward direction.

PWM Pulse	Measure 1	Measure 2	Measure 3	Average
0	0	0	0	0
15	0	0	0	0
30	0	0	0	0
45	0	0	0	0
60	845	778,8	897,3	840,37
75	1220	1181	1250	1217
90	1548	1472	1532	1517,33
105	1752	1720	1737	1736,33
120	1955	1932	1930	1939
135	2104	2122	2124	2116,67
150	2314	2364	2325	2334,67
165	2528	2469	2519	2505,33
180	2640	2551	2643	2611,33
195	2773	2644	2863	2760
210	2902	2824	3006	2910,67
225	2996	3005	3140	3047
240	3146	3330	3287	3254,33
255	3472	3549	3518	3513

**Table 4.16** – *RPM X axis motor forward*

PWM Pulse	Measure 1	Measure 2	Measure 3	Average
0	0	0	0	0
15	0	0	0	0
30	0	0	0	0
45	0	0	0	0
60	651,3	575	685,7	637,33
75	1041	1058	1083	1060,67
90	1361	1391	1402	1384,67
105	1671	1678	1666	1671,67
120	1883	1894	1877	1884,67
135	2034	2133	2061	2076
150	2161	2306	2247	2238
165	2313	2485	2374	2390,67
180	2536	2652	2558	2582
195	2690	2807	2793	2763,33
210	2816	2908	2944	2889,33
225	2933	3030	3051	3004,66
240	3190	3166	3205	3187
255	3459	3462	3483	3468

Table 4.17 – *RPM X axis motor backward*

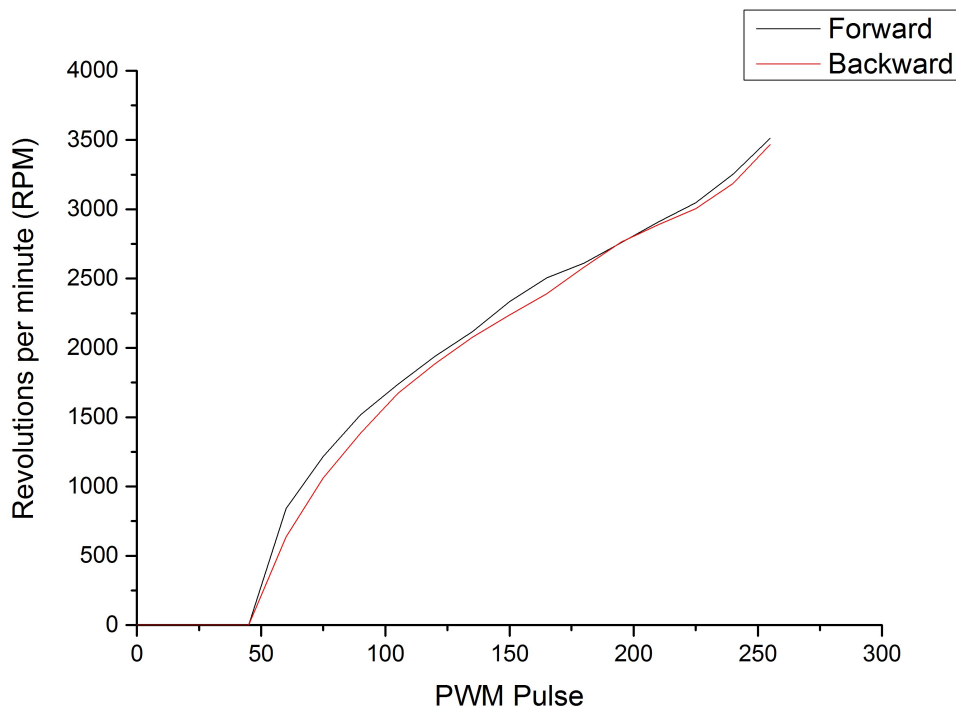


Figure 4.20 – *Revolutions per minute with PWM pulse in X axis motor*

#### 4.2.5.2 Y axis motor

The Y axis motor is responsible for moving the [Testbed](#) in pitch angle. In [table 4.18](#) we can see the correspondence between the [PWM](#) pulse value in the analgical pins of the microcontroller and the voltage measured in the motor terminal for forward and backward direction.

PWM Pulse	Voltage forward (V)	Voltage Backward (V)
0	0	0
15	0,02	0,04
30	0,07	0,12
45	0,56	0,59
60	0,91	0,87
75	1,12	1,09
90	1,36	1,34
105	1,56	1,57
120	1,76	1,73
135	1,9	1,9
150	2,08	2,02
165	2,23	2,23
180	2,4	2,38
195	2,52	2,5
210	2,63	2,6
225	2,9	2,76
240	3,11	3
255	3,35	3,21

**Table 4.18** – Correspondence between voltage and [PWM](#) pulse in Y axis motor

In the same way that in X axis motor, [table 4.19](#) and [4.20](#) shows the [RPM](#) measured for forward and backward direction. [Figure 4.21](#) shows a graphical comparison and we can see that for low [PWM](#) values, the [RPM](#) are higher in backward direction. However, for high [PWM](#) values, the [RPM](#) are higher in forward direction.

PWM Pulse	Measure 1	Measure 2	Measure 3	Average
0	0	0	0	0
15	0	0	0	0
30	0	0	0	0
45	361,4	382,1	392,9	378,8
60	737,6	762,7	789	763,1
75	949	1081	1136	1055,33
90	1254	1322	1376	1317,33
105	1479	1525	1547	1517
120	1660	1745	1749	1718
135	1870	1918	1908	1898,67
150	2273	2284	2305	2287,33
165	2391	2383	2446	2406,67
180	2569	2543	2626	2579,33
195	2662	2649	2766	2692,33
210	2772	2791	2842	2801,67
225	2958	2935	2983	2958,67
240	3164	3174	3165	3167,67
255	3345	3415	3403	3387,67

**Table 4.19** – *RPM Y axis motor forward*

PWM Pulse	Measure 1	Measure 2	Measure 3	Average
0	0	0	0	0
15	0	0	0	0
30	0	0	0	0
45	497,6	523,8	568,8	530,07
60	854,3	890,9	909,5	884,9
75	1103	1162	1164	1143
90	1496	1553	1562	1537
105	1645	1676	1678	1666,33
120	1730	1751	1753	1744,67
135	2150	2166	2186	2167,33
150	2326	2343	2363	2344
165	2418	2436	2450	2434,67
180	2475	2482	2499	2485,33
195	2519	2534	2545	2532,67
210	2726	2850	2856	2810,67
225	2915	2930	2927	2924
240	3075	3056	3055	3062
255	3267	3374	3404	3348,33

Table 4.20 – *RPM Y axis motor backward*

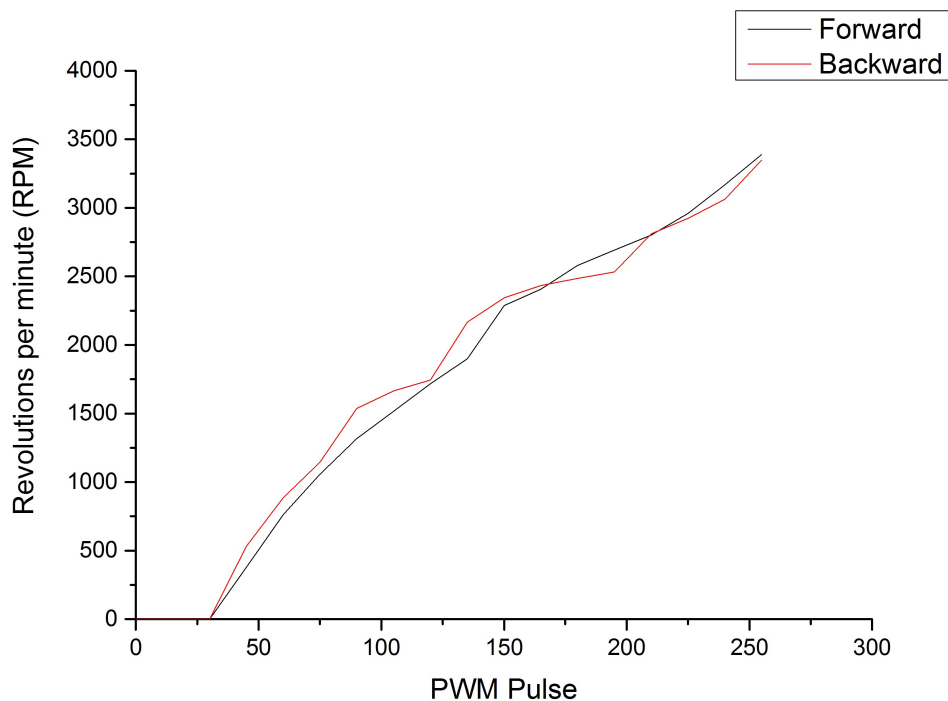


Figure 4.21 – *Revolutions per minute with PWM pulse in Y axis motor*



### 4.2.5.3 Z axis motor

The Z axis motor is responsible for moving the Testbed in yaw angle. In table 4.21 we can see the correspondence between the PWM pulse value in the analgical pins of the microcontroller and the voltage measured in the motor terminal for forward and backward direction.

PWM Pulse	Voltage forward (V)	Voltage Backward (V)
0	0	0
15	0,02	0,04
30	0,08	0,13
45	0,16	0,23
60	0,25	0,34
75	0,35	0,47
90	0,46	0,6
105	0,58	0,74
120	0,72	0,88
135	0,93	1,03
150	1,14	1,19
165	1,36	1,37
180	1,58	1,58
195	1,8	1,8
210	2,03	2,02
225	2,26	2,25
240	2,48	2,47
255	2,72	2,69

**Table 4.21** – Correspondence between voltage and PWM pulse in Z axis motor

In the same way that the other motors, table 4.22 and 4.23 shows the RPM measured for forward and backward direction. Figure 4.22 shows a graphical comparison. In this case, the RPM are higher in backward direction.

PWM Pulse	Measure 1	Measure 2	Measure 3	Average
0	0	0	0	0
15	0	0	0	0
30	0	0	0	0
45	382,4	309,5	346,7	346,2
60	809,8	822,7	821,9	818,13
75	1171	1136	1137	1148
90	1441	1460	1445	1448,67
105	1723	1732	1735	1730
120	1935	1956	1970	1953,67
135	2159	2190	2194	2181
150	2386	2407	2410	2401
165	2552	2604	2582	2579,33
180	2738	2783	2757	2759,33
195	2886	2932	2940	2919,33
210	3033	3079	3114	3075,33
225	3162	3263	3271	3232
240	3345	3390	3412	3382,33
255	3637	3656	3657	3650

**Table 4.22** – *RPM Z axis motor forward*

PWM Pulse	Measure 1	Measure 2	Measure 3	Average
0	0	0	0	0
15	0	0	0	0
30	0	0	0	0
45	843,7	804	722,3	790
60	1146	1144	1182	1157,33
75	1558	1504	1511	1524,33
90	1777	1789	1766	1777,33
105	2024	2004	1991	2006,33
120	2197	2214	2183	2198
135	2390	2402	2365	2385,67
150	2577	2593	2563	2577,67
165	2796	2778	2741	2771,67
180	2913	2952	2908	2924,33
195	3087	3108	3058	3084,33
210	3226	3256	3195	3225,67
225	3327	3375	3318	3340
240	3437	3458	3427	3440,67
255	3622	3661	3624	3635,67

Table 4.23 – *RPM Z axis motor backward*

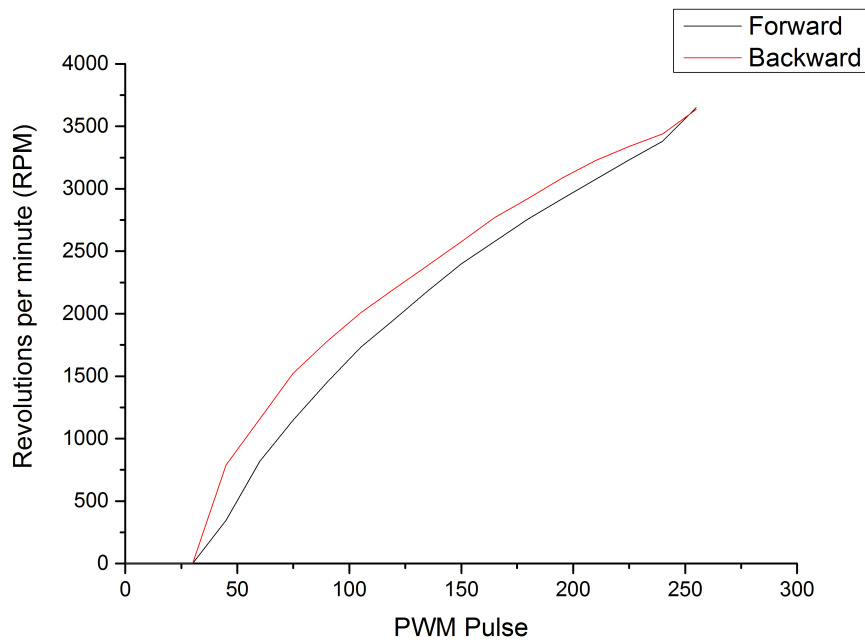


Figure 4.22 – *Revolutions per minute with PWM pulse in Z axis motor*

### 4.2.6 Wireless communication analysis

The [Testbed](#) is thought to be wireless so the communication must be wireless too. For this task it is used the Xbee wireless device from Digi (figure 4.23). This device, which main features can be seen in table 4.24, uses the protocol 802.15.4 (ZigBee) for the wireless communication. We need two of these devices for the communication, one in the [Testbed](#) and another one connected to the ground station.



**Figure 4.23** – Xbee device from Digi [26]

Dimensions (mm)	324.38 x 32.94
Operating Frequency (GHz)	2.4
RF Data Rate (kbps)	250
Indoor Range (m)	Up to 90
Antenna types	Chip, Wire, Whip, UFL, RPSMA
Supply Voltage (V)	2.8-3.4
Serial Interface	3.3V CMOS UART
Logic supply voltage (V)	1.71 to VDD (5)
Temperature range (°C)	-40 to +85
Transmit Current (mA)	150
Receive Current (mA)	55

**Table 4.24** – Xbee specifications [26]

Although when the project will be ended the communication only has to be unidirectional (from the [Testbed](#) to the ground station), during the programming we will need a bidirectional communication between the [Testbed](#) and the ground station in order to test or debug the program. For this reason, we will desing a little protocol that allows us to obtain information and to order to move to the actuators by sending comands.

### 4.3 Additional elements

As it was said in section 2.2, the **Testbed** has to be visually attractive. The microprocessor has pins without any connection, specifically, PD4 and PB0 pins (or digital pin 1 and digital pin 8 in **Arduino**, see figure 4.10) are free. So we will add a **buzzer** in order to give an acoustical output to the **Testbed** and a button for future tasks. A **buzzer** is an electroacoustic transducer that generates a continuous or intermittent noise in the same tone and it is used as signaling mechanism [51]. The **buzzer** will be connected to PD0 pin and it can be seen in figure 4.24.



Figure 4.24 – Buzzer installed in the **Testbed**

Moreover, we will connect a button to PD4 pin (similar to the buttons shown in figure 4.25). This button is a 10 K $\Omega$  pull-up resistor and it can be used for multiple tasks in the **Testbed**.

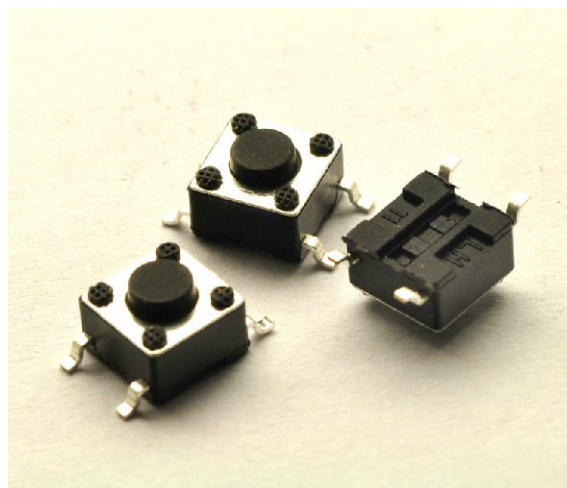


Figure 4.25 – Pull up resistor button [21]

## 4.4 PID controller

In this section we are going to do the same analysis that Victor Burgos did in his bachelor thesis in order to get a better understanding of its implementation in section 4.4.

A **PID** (proportional-integral-derivative) controller is a control loop feedback that is commonly used in applications where an industrial control for an actuator, in our case, the reaction wheels. That algorithm will calculate continuously the difference between the setpoint (target angle in our case) and the measured angle. That controller has the objective to minimize the error between those two values and calculate the new one with the following equation [21] [53]:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (4.4.1)$$

Where  $K_p$ ,  $K_i$  and  $K_d$  all non-negative, denote the coefficients for the proportional, integral, and derivative terms, respectively (sometimes denoted P, I, and D),  $u(t)$  is the output of the controller and the input of the process and  $e(t)$  is the error of the signal treated, so that is the difference between the signal objective  $r(t)$  and the signal that the plant produces  $y(t)$ . Figure 4.26 shows the 4.4.1 equation graphically.

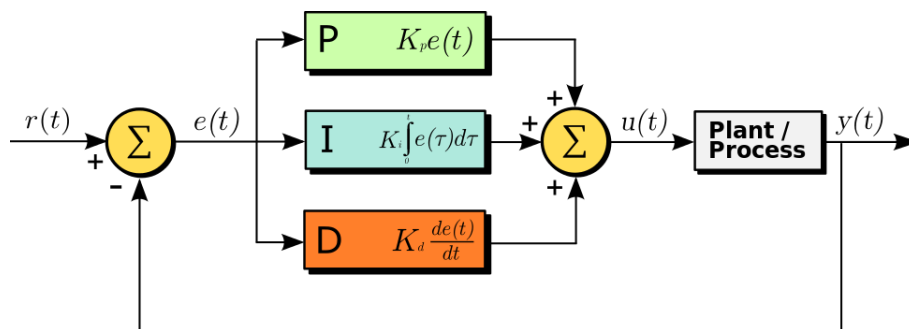


Figure 4.26 – PID block diagram [53]

In the equation 4.4.1 we can see three different terms, one for each term of the PID controller [21] [53]:

- The first block is for the proportional control (P), it is the product between the error signal and the proportional constant, with an error in stationary state even zero.
- The second one is the integral control (I), which decrease and delete the error that proportional block generates. It acts as a deviation between the variable and the target point (integrating and add it to the proportional value).
- Finally the derivative control (D) consider the tendency of the error and allow a fast repercussion of the variable when the process perturbation have been done.

To use the equation 4.4.1, we have to discretize it writing it Z transform:

$$U(z) = K_p E(z) \left[ 1 + \frac{T}{T_i(1-z^{-1})} + T_D \frac{1-z^{-1}}{T} \right] \quad (4.4.2)$$

Where  $T$  is our discrete sample time, we obtained:

$$\frac{U(z)}{E(z)} = a + \frac{b}{1-z^{-1}} + c(1-z^{-1}) \quad (4.4.3)$$

Where  $a = K_p$ ,  $b = \frac{K_p T}{T_i}$  and  $c = \frac{K_p T_D}{T}$

#### 4.4.1 PID tuning

The PID tuning is the election of the parameters  $a$ ,  $b$  and  $c$  described above. Ziegler-Nichols is a technique that allows the PID tuning empirically, without know the equations of the plant, only using the step response of the system (open loop tuning).

Figures 4.27 shows the step response for a process, where  $T_d$  is the time it takes for the system to respond and  $T_1$  is the rise time.

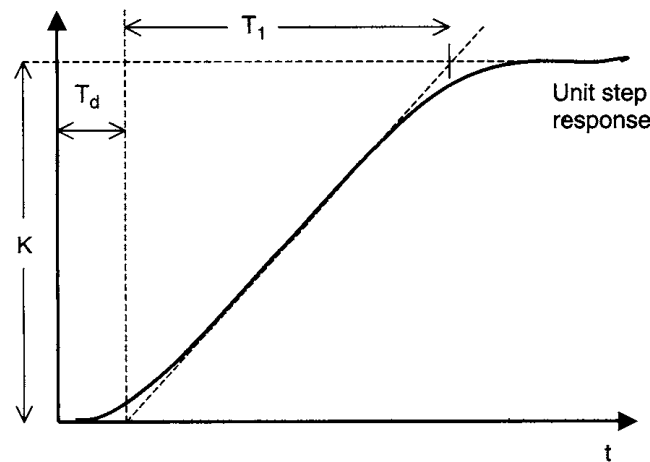


Figure 4.27 – Step response [35]

Once the step response is measured, we can calculate the parameters using the table 4.25, where  $K_s$  is the variation of the step signal and  $K$  is the variation of the system response.

Controller	$K_p$	$T_i$	$T_d$
Proportional (P)	$K_s T_1 / K T_D$		
Proportional + integral (P+I)	$0.9 K_s T_1 / K T_D$	$3.3 T_D$	
Proportional + integral + Derivative (P+I+D)	$1.2 K_s T_1 / K T_D$	$2 T_D$	$0.5 T_D$

**Table 4.25** – Zieger-Nichols parameters [35]

$K_p$  constant is the proportional gain,  $T_i$  is the integral time constant and  $T_d$  is the derivative time constant. If we want to use a PID controller configured with the integral gain ( $K_i$ ) and the derivative gain ( $K_d$ ), we can calculate it with the next expressions:

$$K_i = \frac{K_p}{T_i} \quad (4.4.4)$$

$$K_d = K_p \times T_d \quad (4.4.5)$$

These parameters will be calculated in section 7.6.



4

## CHAPTER

# 5

# SDR'S ACCURACY MEASUREMENT

## 5.1 Previous steps

The main aim of this chapter is the accuracy measurement of the [SDR](#) devices. However, we need to ensure that the signal reception of the device is optimized. Therefore it is necessary to do some previous task like the offset correction (section [5.1.1](#)) and the [SINAD](#) maximization (section [5.1.2](#)).

### 5.1.1 PPM Measurement and offset correction

In picture [3.4](#) can be observed that, although the tune is set to 99,5 MHz, the tune is not situated exactly in the center of the lobe. Another example can be seen in picture [3.5](#), where a 300 MHz signal is input in the [SDR](#) and the tune do not focus in the center of the lobe again. This bad tuning is because it is necessary to do an offset correction. As shows the picture [3.2](#), one of the main part of a [SDR](#) is the tuner. In the case of the [RTL-SDR](#), it uses a tuner with a low quality quartz crystal, with a guaranteed precision of  $\pm 100$  ppm. This means that for each 1 MHz tuning in frequency, the signal will be moved 100 Hz. For example, if we tune to 100 Mhz with a [RTL-SDR](#) with a 64 ppm offset, the final tuning will be 100,064 MHz. Therefore, we have to calculate exactly the ppm deviation to achieve a correct tuning.

**Kalibrate-RTL** is a software that permits to calculate the **ppm** offset. This software uses the signal of the **GSM** mobile phone base station, which has an expensive very high precision atomic clock (0.01 **ppb**) and hence, it generates a well-known frequency signal that allows to calculate the offset deviation of our tuning. In appendix B is described in detail how to use **Kalibrate-RTL** to calculate the offset deviation.

We use that software to calculate the offset deviation of the **RTL-SDR** devices that we will use for accuracy measurement in section 5.2 and we obtain 65,525 **ppm** offset. This value is introduced in **SDR#** (We have to insert 65 due to **SDR#** only accepts integer numbers) and the deviation is corrected as it is shown in the picture below:

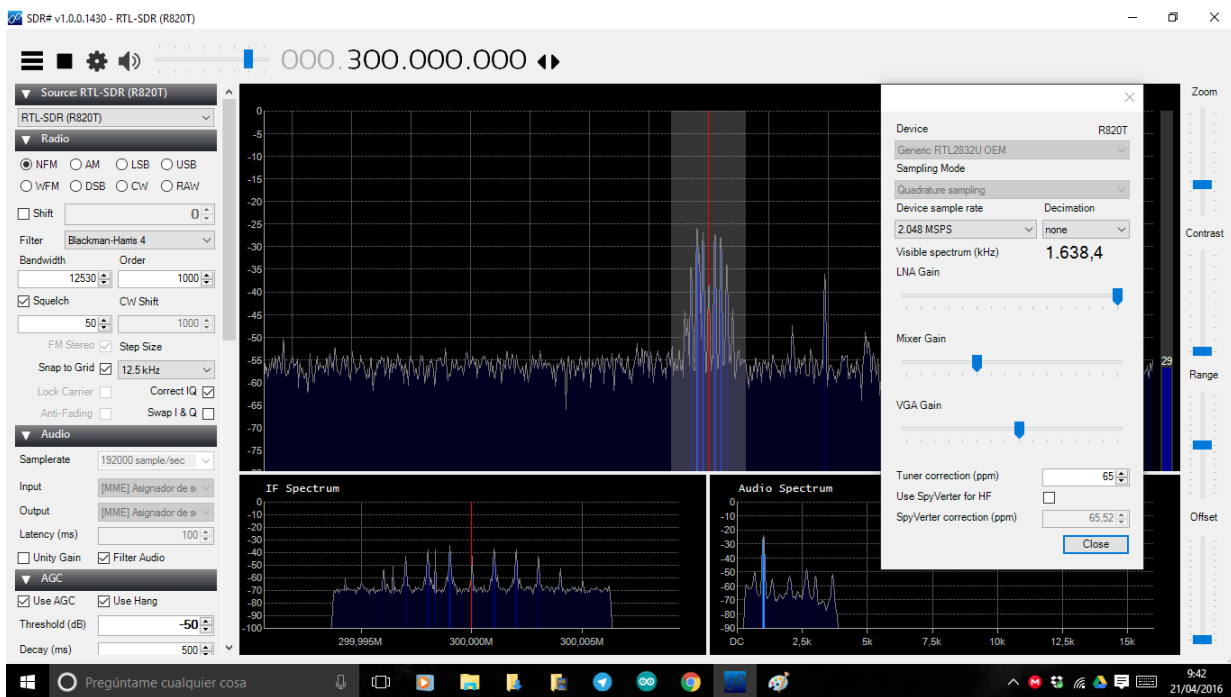


Figure 5.1 – *RTL-SDR tuning with ppm correction.*

It is necessary to do all this process everytime that we use a different **RTL-SDR** device because each one has a different offset deviation. If we do not know exactly the offset, the **RTL-SDR** is practically useless. On the other hand, in the case of **FUNcube Dongle SDR** device, this process of measurement is not necessary because it uses a more quality quartz crystal and the manufacturer of the device guarantees us 1,5 **ppm** offset deviation [30]. Furthermore, it does not exist any tool similar to **Kalibrate-RTL** that allows us to measure the offset, so we have to believe in the manufacturer. Anyway, if we introduce a 1,5 **ppm** offset correction in **SDR#** (With **FUNcube Dongle** is possible to introduce decimal numbers in the ppm correction) the tune is centered correctly as it is shown in picture 3.6.

### 5.1.2 SINAD maximization

Once it has been done the offset correction, the tune is done correctly but the signal reception is still not optimized. We are looking for the best reception as possible with the minimum energy level of transmission or, in other words, we are looking for a high **SINAD** ratio<sup>1</sup> with a low power of transmission.

For this task, we will use a radio communications test set. In our laboratory we have available the Marconi 2995a model (picture 5.3). This device has a signal generator and allows us to generate a signal that is introduced in the **RTL-SDR**. Then, this signal returns to the radio communications test sets and it measures the **SINAD** ratio. Picture 5.2 shows a schematic of the conexions between the Marconi 2995a, the **RTL-SDR** and the laptop, as well as the different connectors used. In the case of the **FUNcube Dongle** the schematic is very similar, with the only difference that it uses a **SMA** connector instead IEC 169-2 connector.

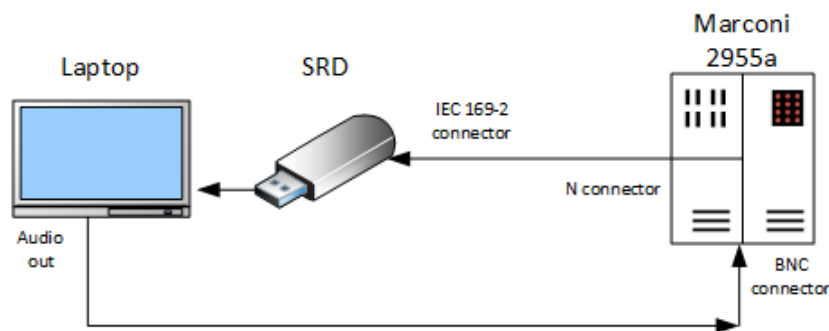


Figure 5.2 – Schematic of the conexions.



Figure 5.3 – Marconi 2995a.

<sup>1</sup>A high **SINAD** ratio means a high quality reception of the signal

In the radio communications test set we generate a 1 kHz frequency<sup>2</sup> signal and we modulate it with a 300 MHz frequency carrier and -100 dBm of energy level. In picture 5.4 can be seen the screen of the Marconi 2955a with this configuration, as well as the SINAD measured. On the other hand, picture 5.5 shows the received signal by the SDR device. On the lower right corner we can observe the audio spectrum of the demodulated signal with a principal lobe in 1 kHz and others secondary lobes in higher frequencies that increases the distortion and, consequently, the SINAD. Obviously, if we could eliminate that secondary lobes, the distortion would decrease. SDR# has the option *Filter Audio* that eliminates all the frequencies higher than 3,3 kHz (Human ear only can perceive frequencies up 3,3 kHz) and consequently, it eliminates most of the secondary lobes of the signal, which increases the SINAD significantly. Picture 5.7 shows how the signal has been filtered, whereas picture 5.6 shows the SINAD measured, which has improved from 9,6 dB to 18,6 dB only filtering the signal.



Figure 5.4 – Screen of the Marconi 2955a without filtering audio.

<sup>2</sup>Marconi 2955a only is able to measure the SINAD of a 1 kHz frequency signal.

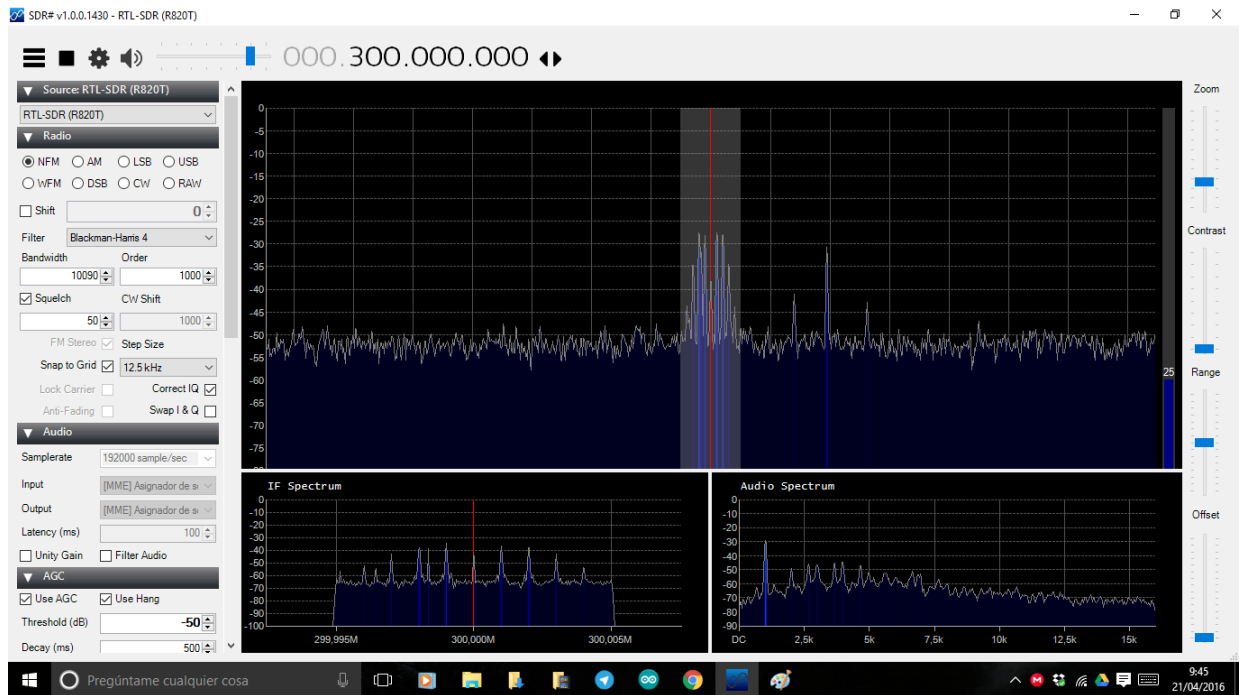


Figure 5.5 – Received signal in SDR#.



Figure 5.6 – Screen of the Marconi 2955a with audio filtering.

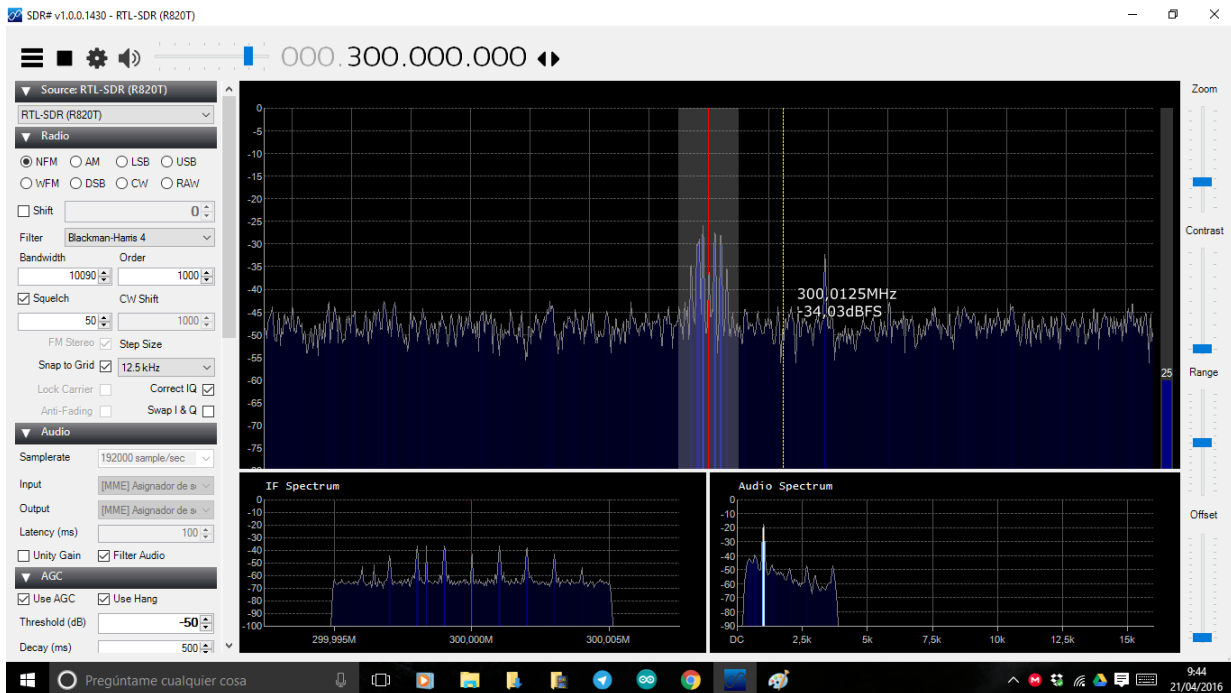


Figure 5.7 – Received signal in SDR# with audio filtering.

In addition, if we want to get the maximum **SINAD** as possible, we can increase the gain of the SDR with SDR#, although we must take into account that increase the gain makes to increase the noise too.

## 5.2 Accuracy measurement

Once we have optimized the signal reception, we finally can do the accuracy measurement, which is the objective of this chapter. It has been considered that a signal with 12 dB **SINAD** is a good signal reception, so we will search the amount of power that we need in the transmission to get 12 dB **SINAD** in the reception. Although both SDR devices have a tunable range until 2 GHz, this measurement will be done in a range of frequency between 0 and 1000 MHz in the case of RTL-SDR and between 0-200 MHz and 450-1000 MHz<sup>3</sup> because the Marconi 2955a can only generate signals until 1 GHz of frequency.[36, Page 1-3]

The communications test set has a GPIB port, so it is possible to program a scrip in MATLAB that will allow us to measure the accuracy automatically. For this task it will be used the KPIB framework, which is used for operating laboratory instruments that are connected to a computer by GPIB or serial connections as the Marconi 2955a communications test set. It is necessary to add in this framework some code including the specific commands of our Marconi in order to make possible the connexion by GPIB port. In [36, Chapter 3-8]

<sup>3</sup>Remember that FUNcube Dongle can not tune between 260 and 410 MHz<sup>3.2</sup>[30].

we can find this commands. It would be desirable that the full measurement process were automatic. However, we must change the tune frequency manually in `SDR#` each time that the script changes the frequency of the carrier. For this reason, the process has to be done in semiautomatic mode.

The measurement process has followed these steps:

- Firstly, the script sets a initial energy level in the signal generator that is different for each frequency (in order to not to wait too much time for the measurement).
- Then, the script increases the energy level in 0.1 dBm.
- Thirdly, the `SINAD` is read. As the reading of the `SINAD` has oscillations, the script does 6 measurements in 2,5 seconds. If three of the six measures are between 11.5 and 13 dB, it is considered that the `Signal to Noise And Distorsion ratio (SINAD)` is 12 dB.
- If the `SINAD` read is not 12 dB, the energy level is increased in 0,1 dBm until read 12 dB.
- Finally, this process is done three times and the final result is the average of them.

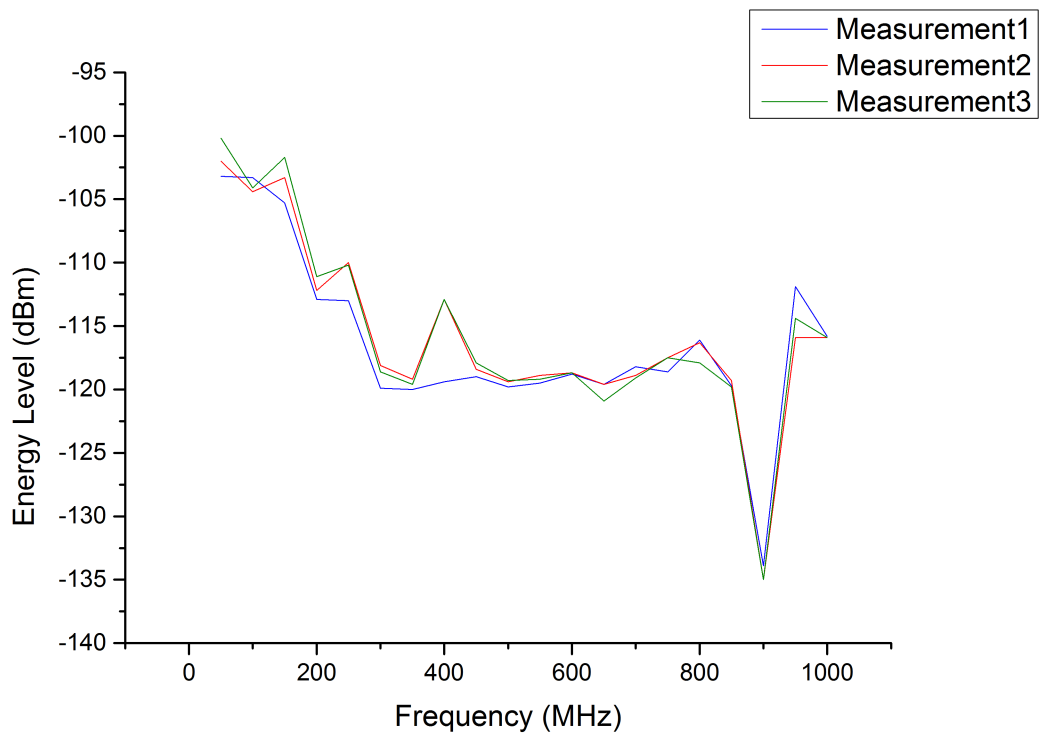
It has been done three measurements because it is very difficult to obtain the same result each time we read the `SINAD`. As we said in section 5.1.1, `RTL-SDR` has a very cheap quartz crystal and, consequently, the `SINAD` read depends very much of the temperature. When the device is running for more than thirty minutes, the `SINAD` measured decreases drastically and it is necessary to wait until the device gets cool. Nevertheless, the `FUNcube Dongle` device has a high quality quartz crystal, so his performance does not depend on the temperature as much as the `RTL-SDR` device.

## 5.2.1 Results

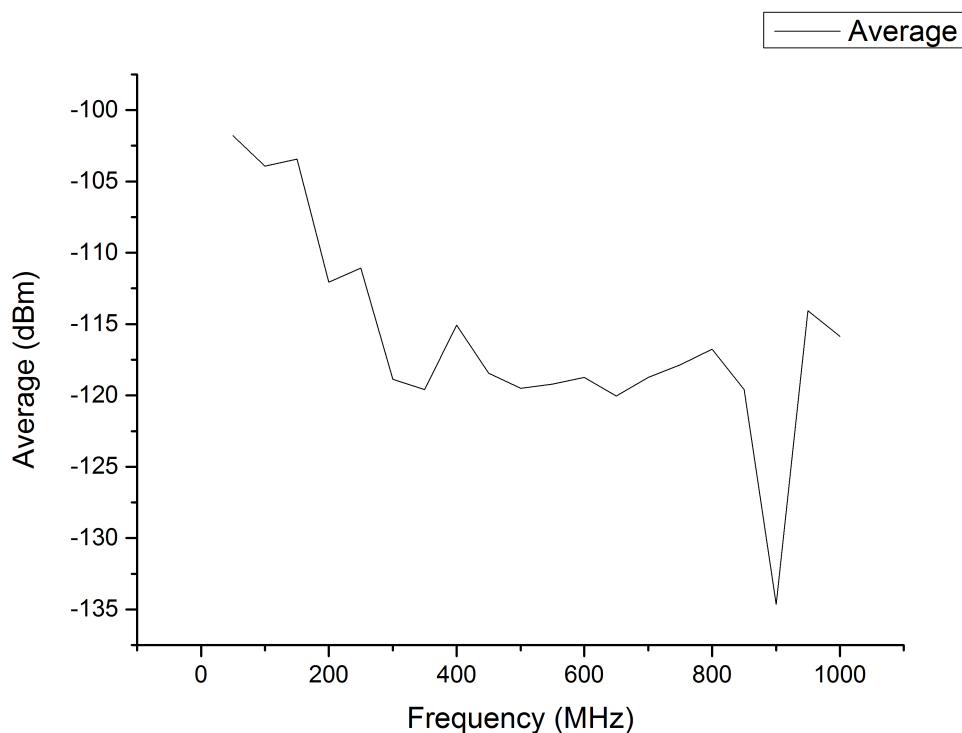
### 5.2.1.1 RTL-SDR accuracy

Graphics 5.8 and 5.9 show the results obtained from the measurement process. In the first one we can see the three measurements done and we can observe the differences between them and the need to do several measures, whereas the second one shows the average of them. We will focus in the last one to obtain conclusions.





**Figure 5.8** – Accuracy of the *RTL-SDR* device.



**Figure 5.9** – Accuracy average of the *RTL-SDR* device.

The *RTL-SDR* has a very irregular behaviour with respect to the frequency. We can distinguish three different stages:

- A first stage between 50 and 350 MHz where the accuracy increases with the frequency (We need less energy level to obtain a 12 dB *SINAD*).
- A second stage between 350 and 800 MHz where, after a little decrease of the accuracy, we obtain a very linear behavior. The accuracy changes very little with the frequency.
- Finally, between 800 and 1000 MHz, we found a peak of accuracy in 900 MHz and then, it becomes linear as in the second stage.

In conclusion, we can say that although the *RTL-SDR* device has a 22-2200 MHz tunable range, its behaviour is not the same in all the range. Specifically, the device reaches its maximum performance at 900 MHz<sup>4</sup>, whereas the worst performance is between 50 and 150 MHz. Unfortunately, *GranaSAT* project has to use the 2 meter (147 MHz) and 70 centimeter (440 MHz) bands<sup>5</sup> for its communications and the *RTL-SDR* do not reach its best performance at these frequencies.

<sup>4</sup>This is the best performance between 50 and 1000 MHz. We do not know the behaviour between 1000 and 2200 MHz.

<sup>5</sup>Bands used for amateur radio community.

### 5.2.1.2 FUNcube accuracy

Graphics 5.10 and 5.11 show the results obtained from the measurement process. Unlike RTL-SDR, we do not observe big differences between the measurements because, as we said before, the FUNcube Dongle has a higher quality quartz and its behaviour does not depend on the temperature as much as the other device. However, the three measures have been done to check this feature. The second graphic shows the average of the measurements and it will be used to obtain the final conclusions.

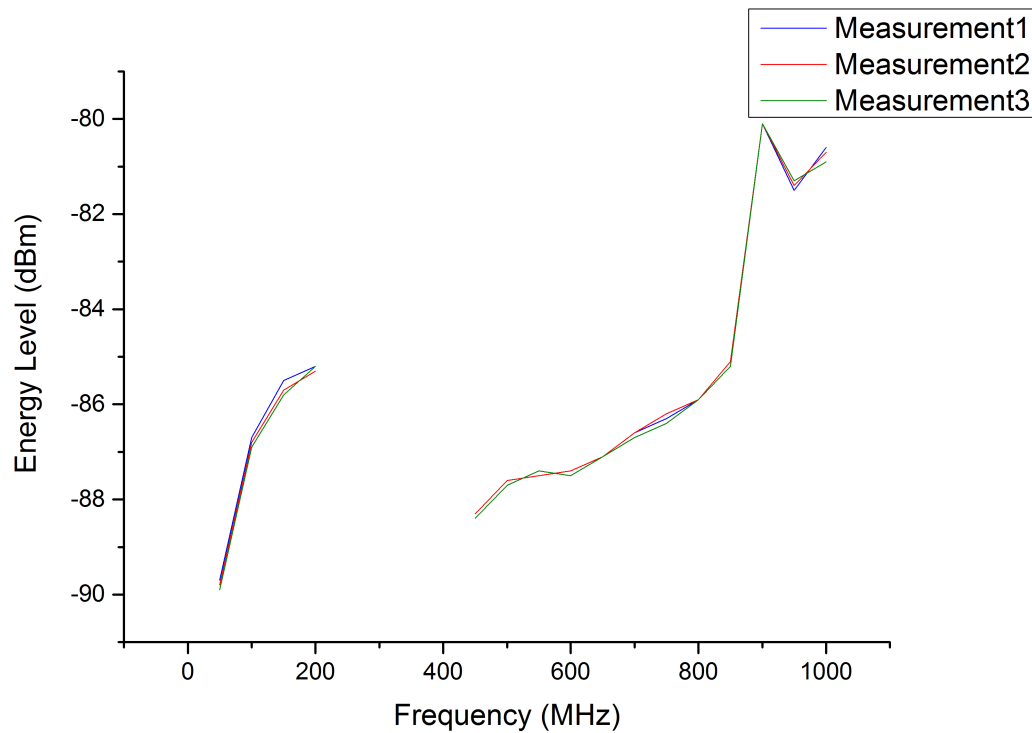
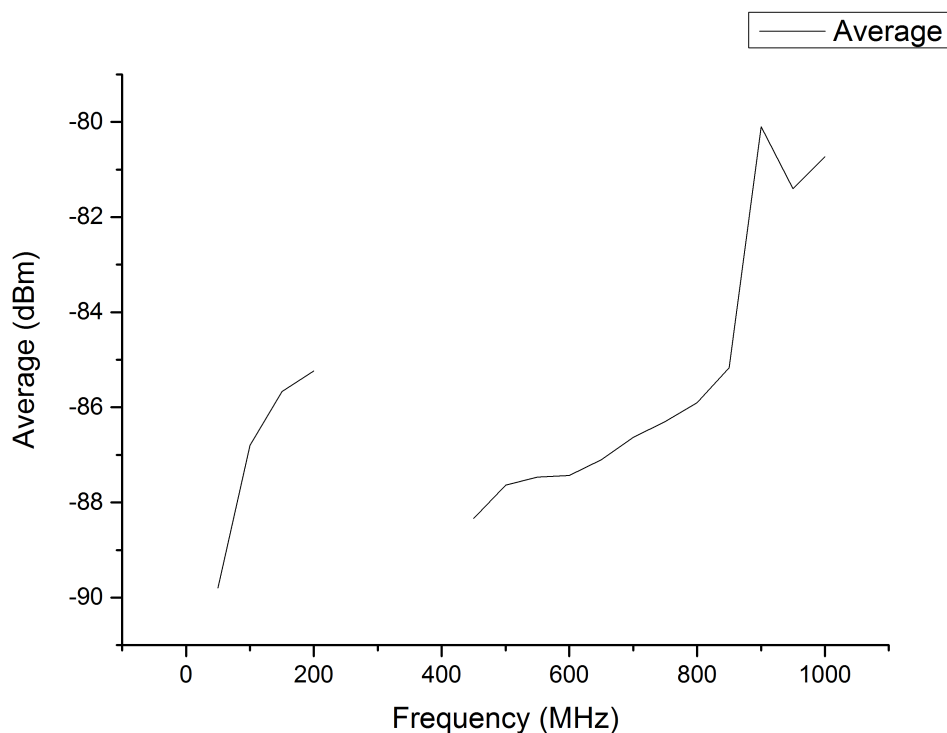


Figure 5.10 – Accuracy of the FUNcube Dongle device.

+



**Figure 5.11** – Accuracy average of the *FUNcube Dongle* device.

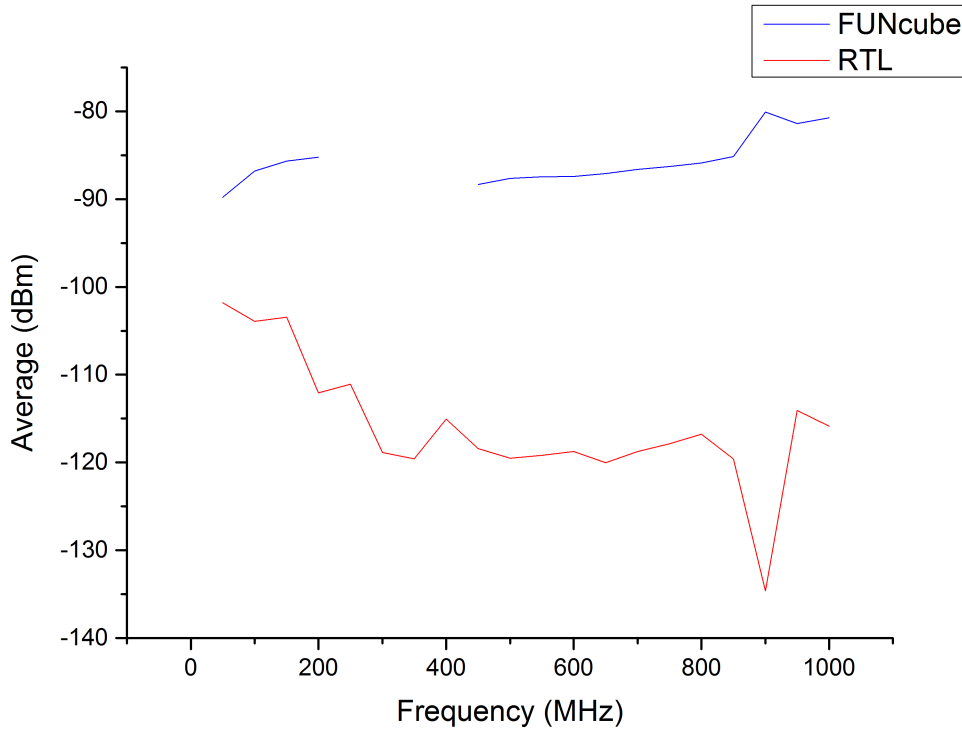
If we compare graphic 5.11 with graphic 5.9 we can observe that, although the *FUNcube Dongle* device is theoretically more accurate than *RTL-SDR* device, the last one needs less energy level to obtain 12 dB *SINAD*. This is because *SDR#* has a plugin that allows increase the gain of the *RTL-SDR* and there is not any similar plugin for the *FUNcube Dongle*. If we measured the accuracy of both devices in the same conditions, the *RTL-SDR* would need more energy level to obtain 12 dB *SINAD*.

Analyzing the graphics we can observe that, in general, the accuracy of the device decreases when we increase the frequency. We can observe others three stages:

- A first stage between 50 and 200 MHz where the accuracy decreases with the frequency. We found the optimal point at 50 MHz.
- A second stage between 200 and 450 MHz where there is no accuracy because the *FUNcube Dongle* does not tune between 240 and 420 MHz.<sup>6</sup>
- Finally, between 450 and 1000 MHz, we found a behaviour similar to the first stage, where the accuracy decreases with the frequency. Moreover, there is a peak of low accuracy at 900 MHz.

<sup>6</sup>The measures has been done each 50 mHz.

Curiously, the [RTL-SDR](#) reaches its best performance at 900 MHz, whereas the [FUNcube Dongle](#) reaches its worst performance at the same frequency. However, if we compare the required energy level in both devices to obtain 12 dB [SINAD](#), we can observe that the [FUNcube Dongle](#) has a more linear behaviour than the [RTL-SDR](#) as it shows the graphic 5.12. The reason is the higher quality quartz of the first one again.



**Figure 5.12** – Comparison between *RTL-SDR* and *FUNcube Dongle* accuracy.

All this process has been published in [GitHub.com](#), including the [SDR](#) analysis, the [MATLAB](#) code and the results. It can be downloaded from [https://github.com/granasat/SDR-Research-Development/tree/master/SDR\\_Sensibility\\_measurement](https://github.com/granasat/SDR-Research-Development/tree/master/SDR_Sensibility_measurement).

## CHAPTER

# 6

# POWER SUPPLY DESIGN AND IMPLEMENTATION

As it was said in [4.2.3](#), we have to replace the [Testbed](#)'s power supply for a new one. There are components, like the battery, that we can reuse but most of them are useless or we do not have it, like the step up or the charger module. We said in [4.2.3](#) that we are going to design the step up module in order to learn how to desing [PCBs](#), although at the end we will buy the step up instead of built it. The charger module will be bought directly without designing it previously.

In this chapter we will describe each element of the power supply in depth and, in the case of the step up module, we will describe its design process too. In section [6.5](#) we will describe the final assembly.

## 6.1 Battery

The battery used is the model [MGL2803](#) from Enix Energies, that was chosen by Victor Burgos Gonzalez [[21](#)]. The main characteristics are shown in table [6.1](#).



**Figure 6.1** – Battery *MGL2803*

Nominal voltage (V)	3.75
Nominal capacity 20°C (Ah)	6.8
Technology	Rechargeable Lithium Ion Battery Pack
Discharge Cut off V Nominal (V)	2.8
Maximum recommended discharge current (A)	5
Charge Voltage Nominal (V)	4.2
Charge Termination Current (mA)	150
Discharge temperature range (°C)	-10 to +60

**Table 6.1** – ENIX *MGL2803* Battery Specifications [27]

The battery is connected with the step up module and with the charging module, with switches between the connections that allow us turn on and turn off the [Testbed](#).

# 6

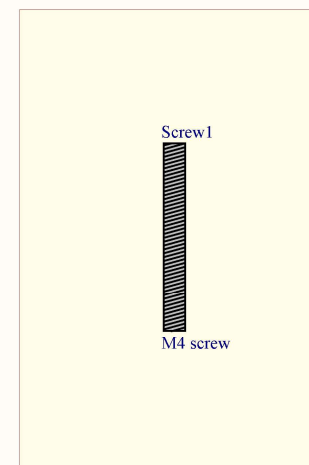
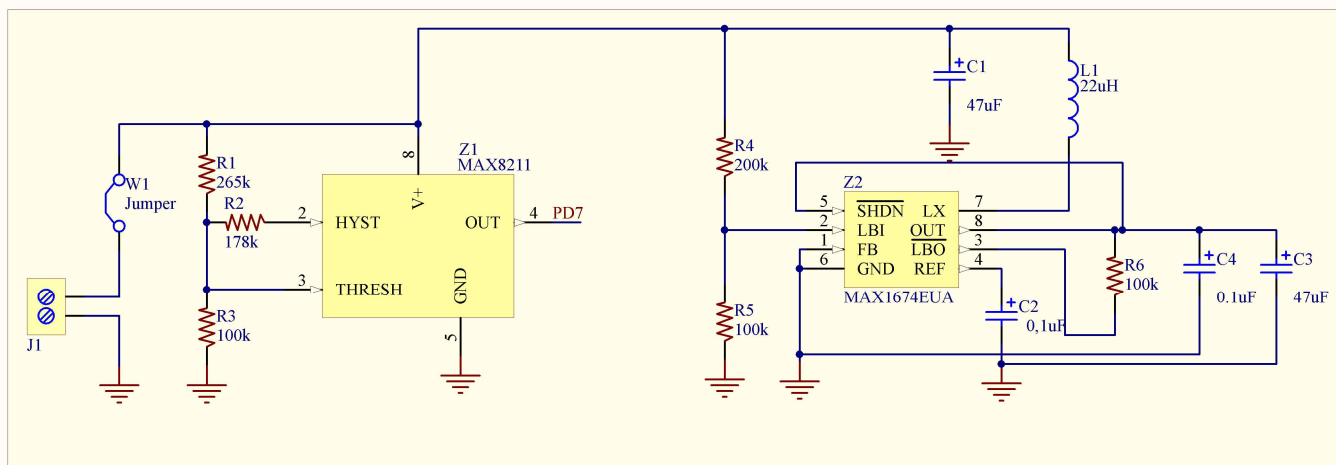
## 6.2 Step Up


### 6.2.1 Step Up design

The design of the power supply has been done in [Altium Designer 15](#), a [PCB](#) design tool which includes the schematics design, [PCB](#) design and 3D model of the [PCB](#). The [following page](#) shows the schematic of the power supply. As we can observe, the design is based in two main devices: [MAX1674](#) and [MAX8211](#), which will be described below. Both devices are manufactured by Maxim Integrated, a company that allows us to ask for free samples. Hence, we have got this devices at zero cost.

POWER SUPPLY

MECHANICAL SUPPORT

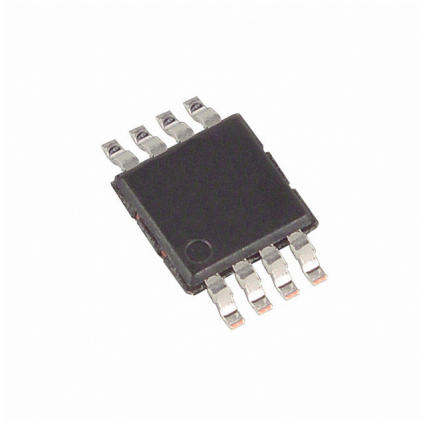


Designer's signature	Sheet title: <b>Power System Schematic</b>		Dpto. Electrónica y Tecnología de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain St. Andrés Roldán Aranda
	Project title: <b>Power_System_V1.PrjPcb</b>		
Supervisor's signature	Desginer: <b>Antonio José Ortiz Sánchez</b>		
	Date: <b>20/01/2016</b>	Revision: <b>V4</b>	

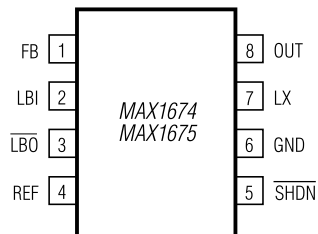


### 6.2.1.1 MAX1674

**MAX1674** (Figure 6.2) is a compact, high-efficiency, step-up DC-DC converters fit in small  $\mu$ MAX packages. The input voltage ranges from 0.7 V to  $V_{OUT}$ , where  $V_{OUT}$  can be set from 2 V to 5.5 V. Moreover, it has a preset, pin-selectable output for 5 V or 3.3 V [38]. Thus, this device is perfect for our power supply because we need to increase the valtage from 3,75 V to 5 V. Figure 6.3 shows the pin out of the device.



**Figure 6.2** – *MAX1674* device



**Figure 6.3** – *MAX1674* PinOut

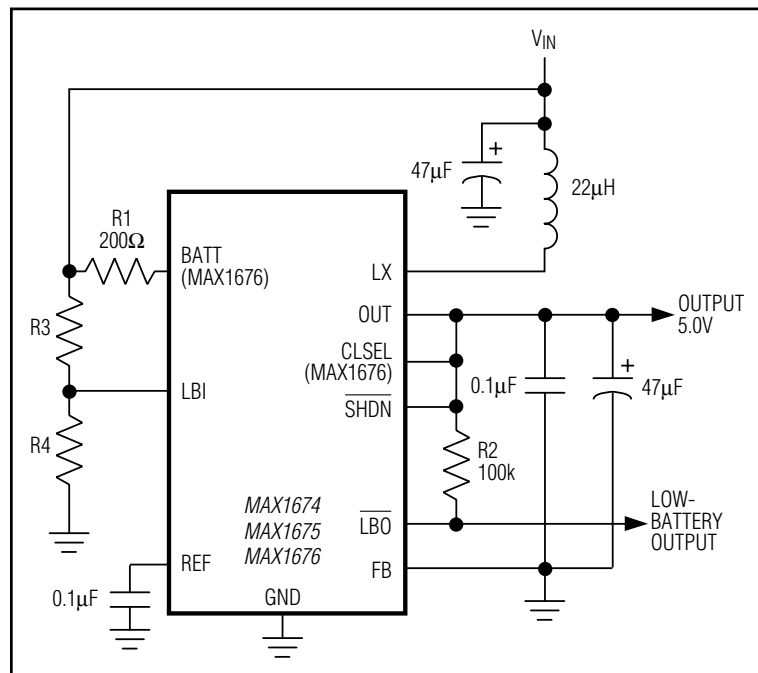
The picture 6.4 shows the typical operating circuit of the **MAX1674** for a preset output voltage of 5 V<sup>1</sup>. **MAX1674** has a low-battery detector, an on-chip comparator for low-battery detection. If the valtage at LBI pin falls below the internal reference voltage (1,30 V), LBO pin sinks current to GND to avoid damages in the device. Since LBI current is less than 50 nA, large resistor values ( $R_4 \geq 260 \text{ k}\Omega$ ) can be used to minimize loading of the input supply. R3 can be calculated using the equation 6.2.1 for  $V_{TRIP} \geq 1,3 \text{ V}$ .  $V_{TRIP}$  is the level where the low-battery detector output goes low, and  $V_{REF}$  is the internal 1.30 V reference [38].

$$R3 = R4 \times [(V_{TRIP}/V_{REF}) - 1] \quad (6.2.1)$$

<sup>1</sup>Resistors R3 and R4 correspond with resistors R4 and R5 respectively in the power supply schematic.

Setting  $R4 = 120\text{ k}\Omega$  and  $V_{TRIP} = 3,7\text{ V}$  (we are using a 3,75 V battery), we obtain  $R3 = 221,538\ \Omega$  (equation 6.2.2). However, we have to approximate this value to  $R3 \cong 220\text{ k}\Omega$  in order to use commercial values' resistors.

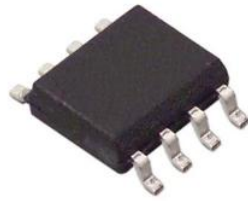
$$R3 = 100\text{ k} \times [(3,7/1,3) - 1] = 184,615\ \Omega \cong 200\ \Omega \quad (6.2.2)$$



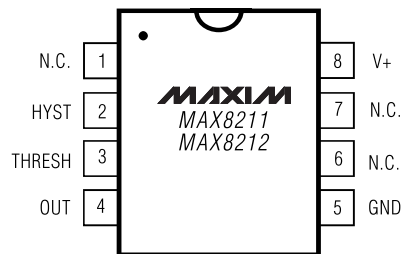
**Figure 6.4** – *MAX1674* typical operating circuit for 5 V output

### 6.2.1.2 MAX8211

**MAX8211** (Figure 6.5) is a CMOS micropower voltage detectors that warn microprocessors ( $\mu\text{Ps}$ ) of power failures [39]. We will use this device to switch off the system when the battery runs out and avoids damages in the system and in the battery. Figure 6.5 shows the pin out of the device.



**Figure 6.5** – *MAX8211* device



**Figure 6.6** – *MAX8211 PinOut*

The picture 6.7 shows the typical operating circuit for both undervoltage and overvoltage detection<sup>2</sup>. To ensure noise-free output switching, hysteresis is frequently used in voltage detectors. For *MAX8211* the *HYST* output is on for voltages greater than 1.15 V. *R3* controls the amount of current supplied from the *HYST* output to the mid-point of the resistor divider, and hence, the magnitude of the hysteresis. *R1* tends to have a value between 10  $k\Omega$  and 10  $M\Omega$ , while *R2* and *R3* can be calculated with equations 6.2.3 and 6.2.4 respectively [39].

$$R2 = R1 \times \frac{(V_U - V_{TH})}{V_{TH}} \quad (6.2.3)$$

$$R3 = R2 \times \frac{(V_L - V_{TH})}{(V_U - V_L)} \quad (6.2.4)$$

Where  $V_{TH}$  is 1,15 V,  $V_U$  is the desired upper trip point and  $V_L$  is the lower trip point. These values must be chosen according to the characteristics of our battery 6.1. In our case, we set  $V_L = 3,6$  V and  $V_U = 5$  V. If we set  $R1 = 100$   $k\Omega$  and we solve equations 6.2.3 and 6.2.4, we obtain  $R2 = 330$   $k\Omega$  (Equation 6.2.5) and  $R3 = 560$   $k\Omega$  (Equation 6.2.6) after

<sup>2</sup>Resistors *R1*, *R2* and *R3* correspond with resistors *R3*, *R1* and *R2* respectively in the power supply schematic.

round the results to commercial values.

$$R2 = 100 \text{ k} \times \frac{(5 - 3,6)}{1,15} \cong 330 \text{ k}\Omega \quad (6.2.5)$$

$$R3 = 330 \text{ k} \times \frac{(3,6 - 1,15)}{(5 - V_{3,6})} \cong 560 \text{ k}\Omega \quad (6.2.6)$$

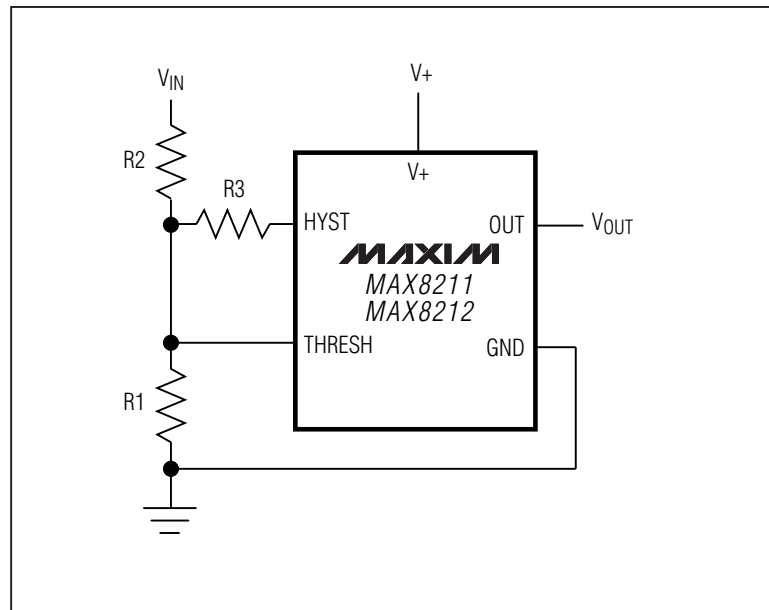
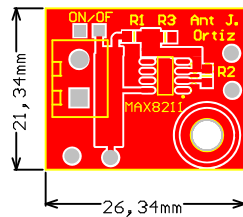


Figure 6.7 – *MAX8211* typical operating circuit

### 6.2.2 2D view of the PCB design

The following pages show the 2D view of the PCB in both layers, [top](#) and [bottom](#). Due to the PCB has small dimensions, it has been included a scaled image of the PCB four times bigger, in order to the reader could watch the details easily.

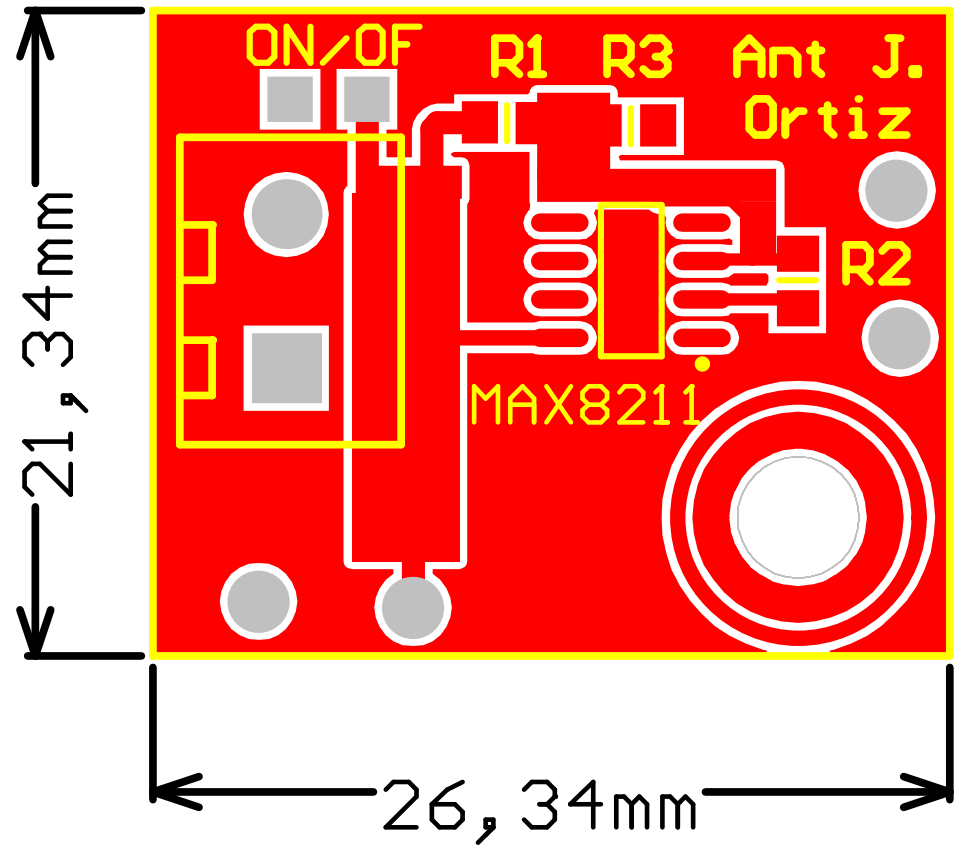
# Power System PCB



Scale 1:1

## Description:

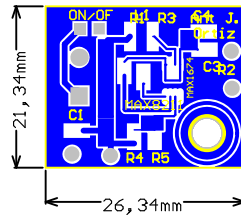
Step up power system that allows to raise the voltage from 3,75 U to 5 U.



Scale 4:1

Designer's signature:	Sheet title: Power System PCB Top Layer	Dpto. Electronica y Tecnologia de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda	
	Project title: GranaSAT		
Supervisor's signature:	Designer: Antonio Jose Ortiz Sanchez	Dates: 04/07/2016 Revisions: 1	
	Supervisor: Andres Roldan Aranda	Sheet 1 of 2	

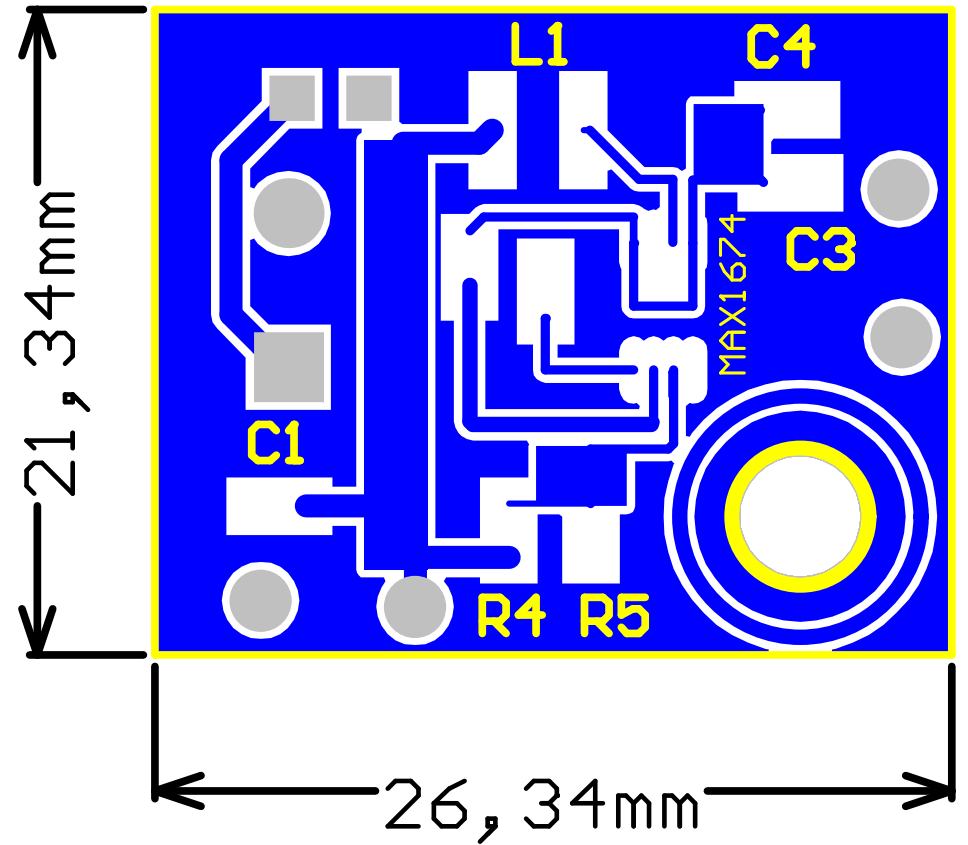
# Power System PCB



Scale 1:1

## Description:

Step up power system that allows to raise the voltage from 3,75 U to 5 U.

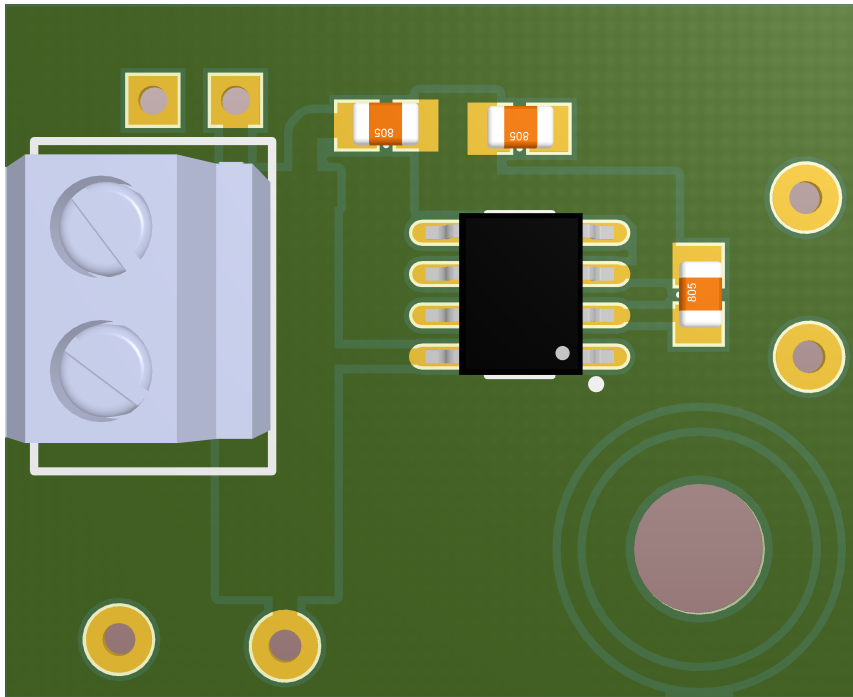


Scale 4:1

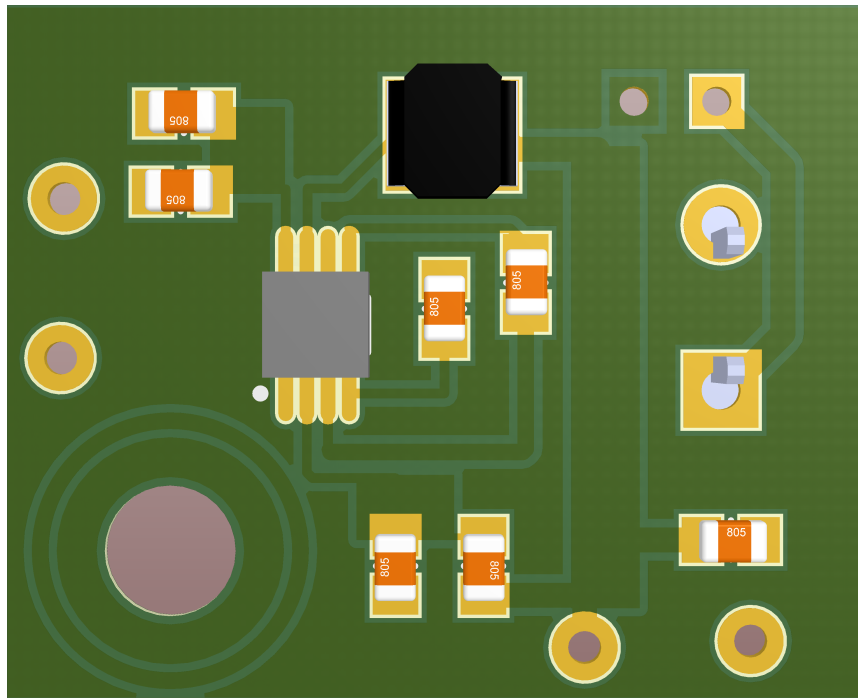
Designer's signature:	Sheet title: Power System PCB Bottom Layer	Dpto. Electronica y Tecnologia de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda	
	Project title: GranaSAT		
Supervisor's signature:	Designer: Antonio Jose Ortiz Sanchez	Dates: 04/07/2016 Revisions: 1	
	Supervisor: Andres Roldan Aranda	Sheet 2 of 2	

### 6.2.3 3D view of the PCB design

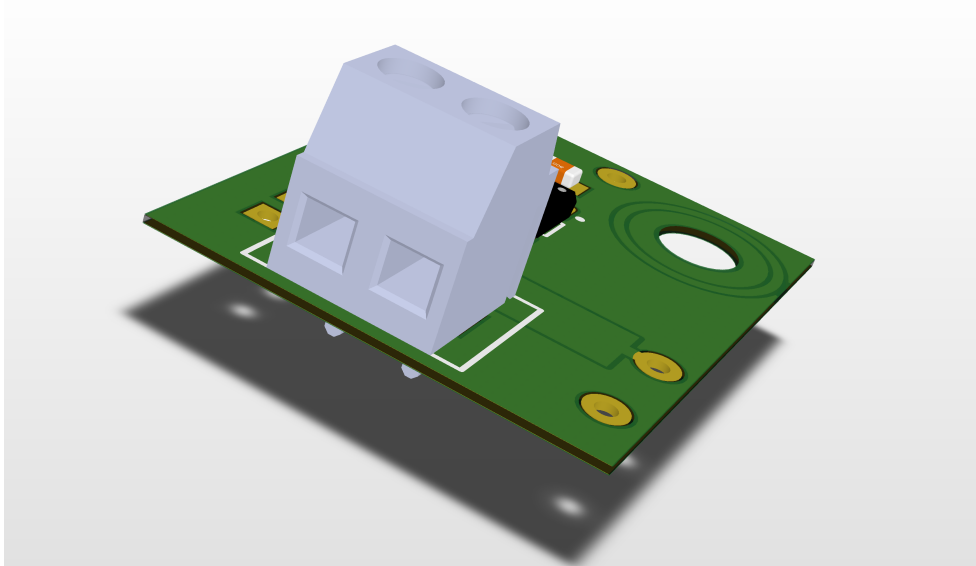
Figures 6.8 to 6.13 show the 3D model of the designed PCB since different points of view. This 3D model can be used as reference in order to have an idea of the real size of the components and the final appearance of our PCB. The 3D models of each component have been downloaded from [23].



**Figure 6.8** – *3D view of the top side*

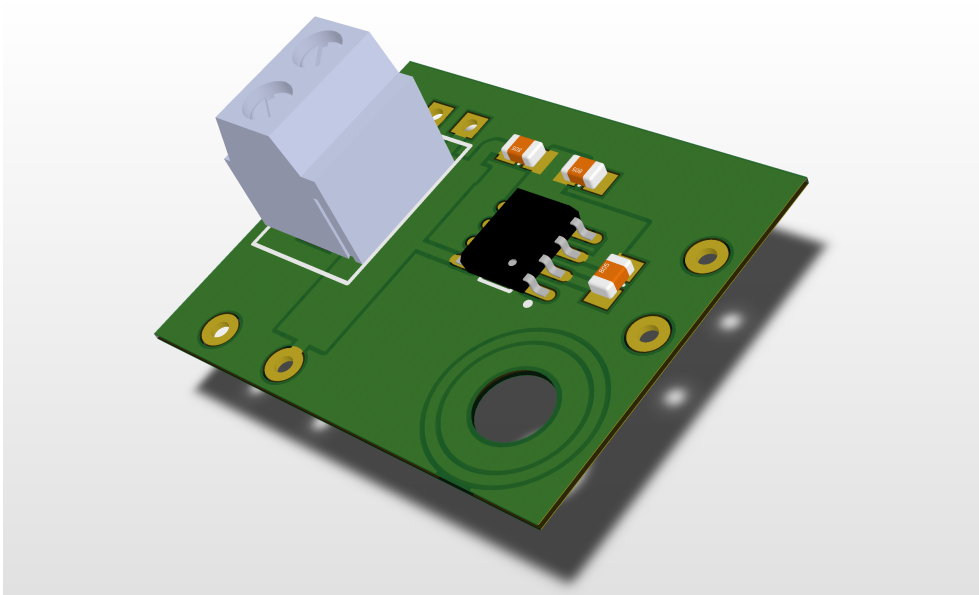


**Figure 6.9** – 3D view of the bottom side

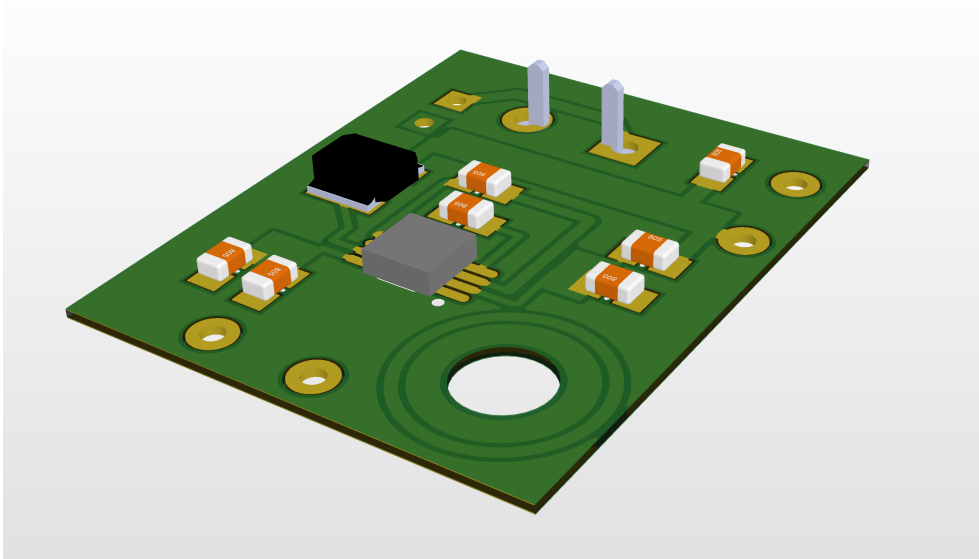


**Figure 6.10** – 3D view of the top left side

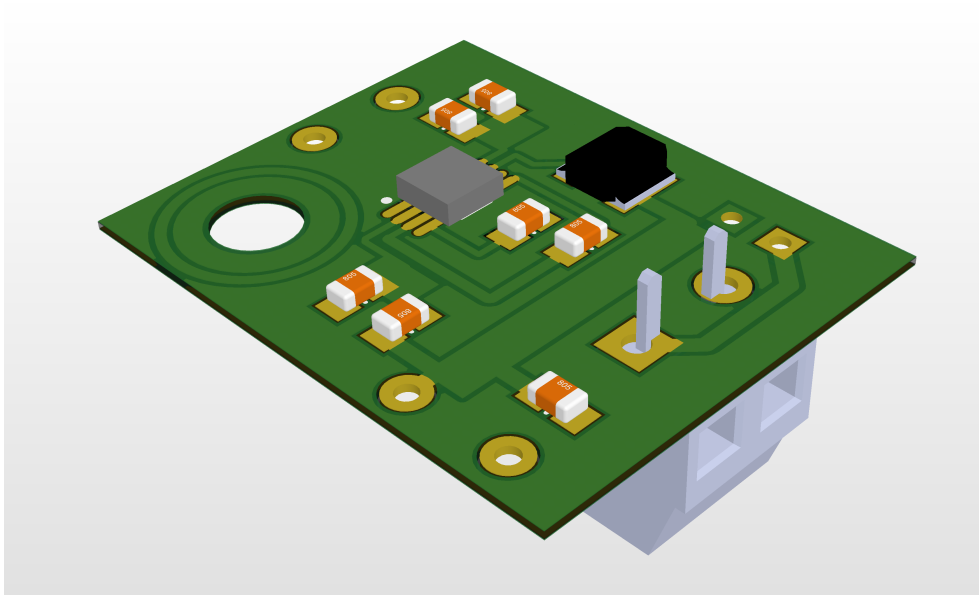




**Figure 6.11** – *3D view of the top right side*

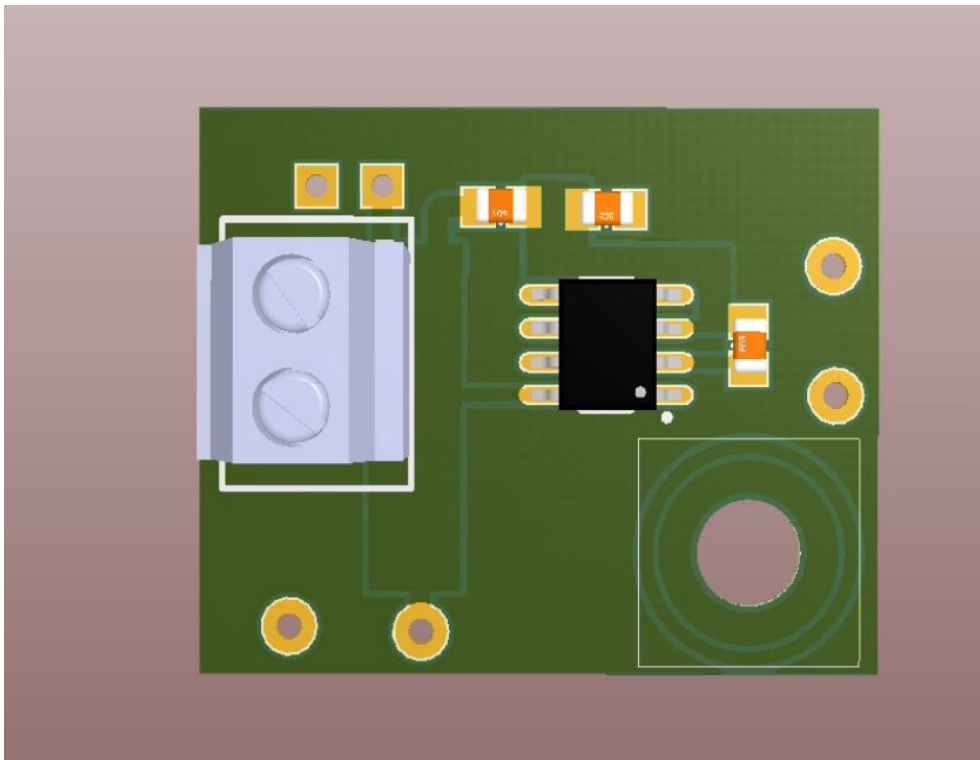


**Figure 6.12** – *3D view of the bottom left side*



**Figure 6.13** – *3D view of the bottom right side*

The following video shows the 3D view of the [PCB](#).



**Video 6.1** – 3D view of the PCB

If there is any problem to see the video correctly, see [https://www.youtube.com/watch?v=hu8IBr5nM34&feature=youtu.be&ab\\_channel=GranaSATTeam](https://www.youtube.com/watch?v=hu8IBr5nM34&feature=youtu.be&ab_channel=GranaSATTeam).

#### 6.2.4 Step Up final implementation

As we said at the beginning of this chapter, although we have design the step up PCB, finally we decided to buy it instead of manufacturing it because it was much cheaper. The step up chosen has been the MT3608. We can buy a little PCB with the step up and the typical operating circuit, ready for plug and play (figure 6.14). This PCB has a potentiometer that allows us to change the output voltage turning the screw. We will turn the screw until the output voltage will be 5 V.

Figure 6.15 shows the typical operating circuit of the device. This circuit is very similar to the circuit of the step up PCB 6.14, except for the potentiometer.

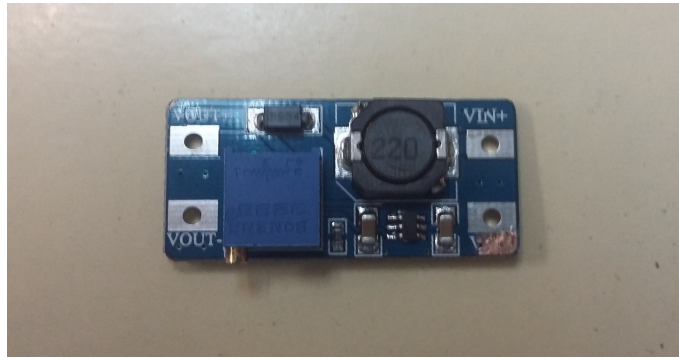


Figure 6.14 – MT3608 2A Max DC-DC Step Up

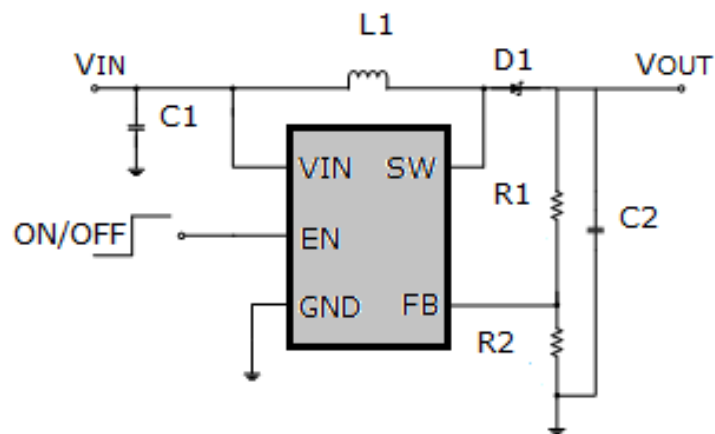
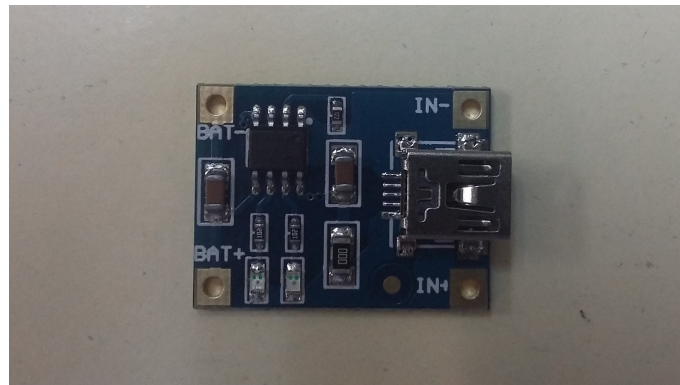


Figure 6.15 – MT3608 typical operating circuit [15]

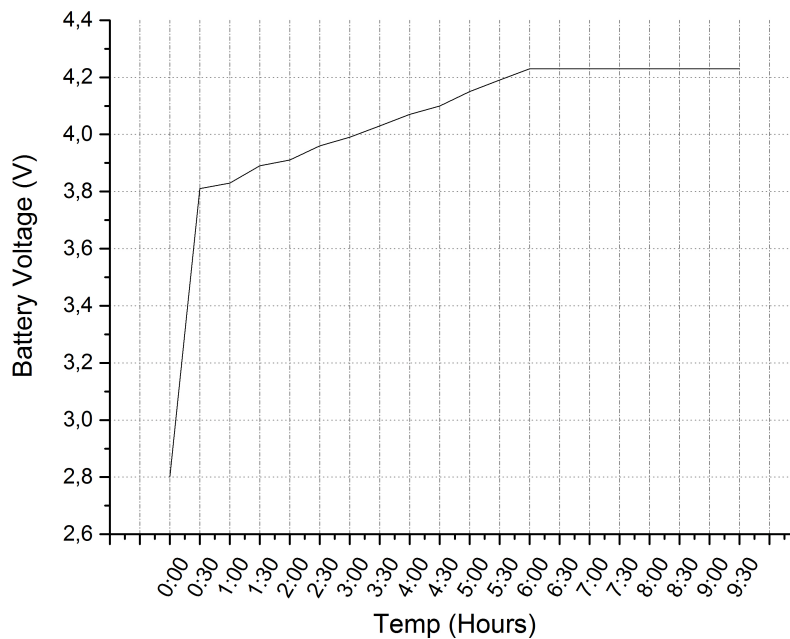
## 6.3 Charger Module

The model used is the [TP4056](#), from NanJing Top Power ASIC Corp. This charger is a complete constant-current/constant-voltage linear charger for single cell lithium-ion batteries as [MGL2803](#), which makes the TP4056 ideally suited for portable applications. Furthermore, the charger can work within [USB](#) and wall adapter [24].



**Figure 6.16** – TP4056 Li-Ion battery charger

In order to know how the behaviour of the battery charger is, the battery has been completely discharged and charged. The voltage between the battery terminals has been measured each thirty minutes. The plotted results are shown in figure 6.17.



**Figure 6.17** – Battery charger behaviour

The battery has been charging during 9 hours and 30 minutes. We can know that the battery is completely charged when the green led goes on. The charger module has another additional red led that, together the green led, give us additional information. Table 6.2 shows the information given by the light indicators.

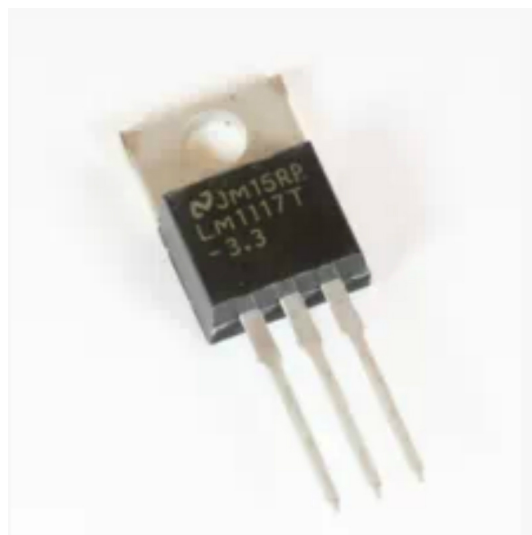
Charge state	Red LED $\overline{\text{CHRG}}$	Green LED $\overline{\text{STDBY}}$
charging	bright	extinguish
Charge Termination	extinguish	bright
Vin too low; Temperature of battery too low or too high; no battery	extinguish	extinguish
BAT PIN Connect 10u Capacitance; No battery	Green LED bright, Red LED Coruscate T=1-4 S	

**Table 6.2** – Indicator light state [24]

## 6.4 Voltage Regulator

Although it is necessary 5 V to power the microprocessor, the XBee wireless module is powered with 3.3 V. Due to our step up give an output of 5 V, we will use a voltage regulator connected to the step up output to obtain 3.3 V. Specifically, we will use the LM1117 regulator (3.3 V version, with NDE Package 3-Pin TO-220). The LM1117 is a low dropout voltage regulator manufactured by Texas Instruments with a dropout of 1.2 V at 800 mA of load current. It is available in five fixed voltages, 1.8 V, 2.5 V, 3.3 V, and 5 V and it offers current limiting and thermal shutdown [37].

Figure 6.18 shows the LM1117 regulator and the figure 6.19 shows its pin out. In our case, we will have 5 V input and 3.3 V output.



**Figure 6.18** – LM1117 Regulator [18]

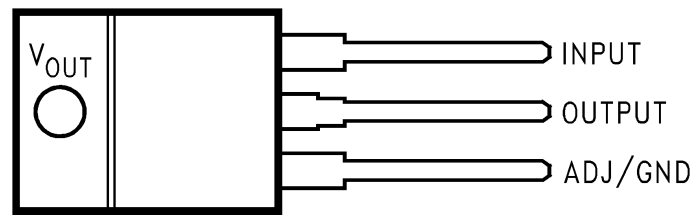


Figure 6.19 – LM1117 PinOut [37]

## 6.5 Final implementation

Once all the components of our power system are ready, we have to do the connections and weld the whole system to the [PCB Testbed](#). First, we will connect the battery with the [TP4056](#) charger and the [MT3608](#) step up using female and male headers (figure 6.20 in order to plug and unplug the battery whenever we want).

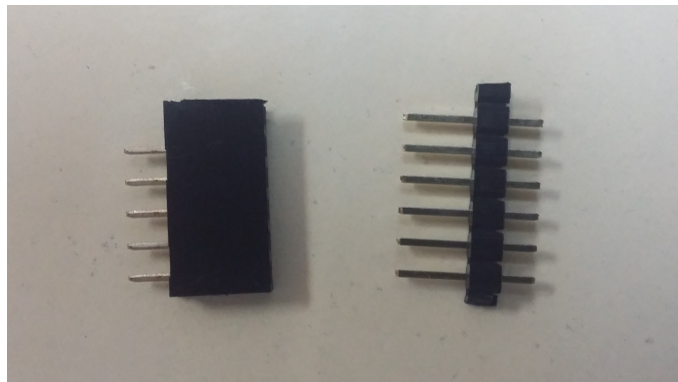


Figure 6.20 – Female and male headers

# 6

Additionally, we will use the two switches of the figure 6.21. One of them will be connected between the battery and the step up. This one switch on and switch off the power supply of the [PCB](#). The other one is connected between the battery and the charger and allows the battery to charge if it is switch on. Thanks to the switches, it is possible to power the [Testbed](#) directly from the charger without using the battery, even if the battery is disconnected. Figure 6.22 shows the battery connected with the charger and the step up.

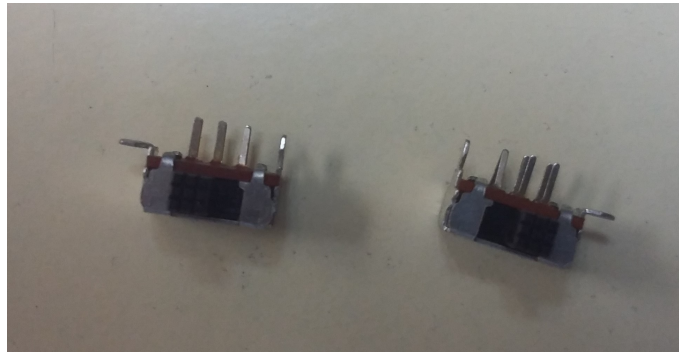


Figure 6.21 – Switches



Figure 6.22 – System power before adding it to the PCB

Before welding the power system to the PCB, we have to connect the LM1117 voltage regulator to the output of the step up. Now we have 5 V in the output of the step up to power the microprocessor and 3.3 V in the output of the voltage regulator to power the XBee wireless module. Figure 6.23 shows the output of the voltage regulator wired with the Vcc pin of the XBee module (Blue wire) and the output of the step up wired with the Vcc pins of the microprocessor and of the motors drivers (Red wire).



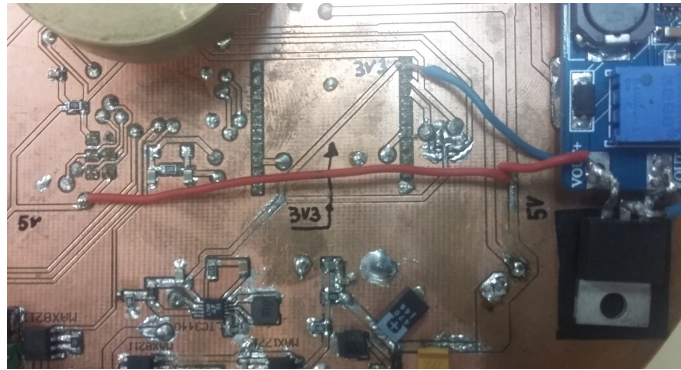


Figure 6.23 – Regulator output

Finally, we can weld the whole power system to the PCB. Figure 6.24 shows graphically each component of the power system with its respective label.

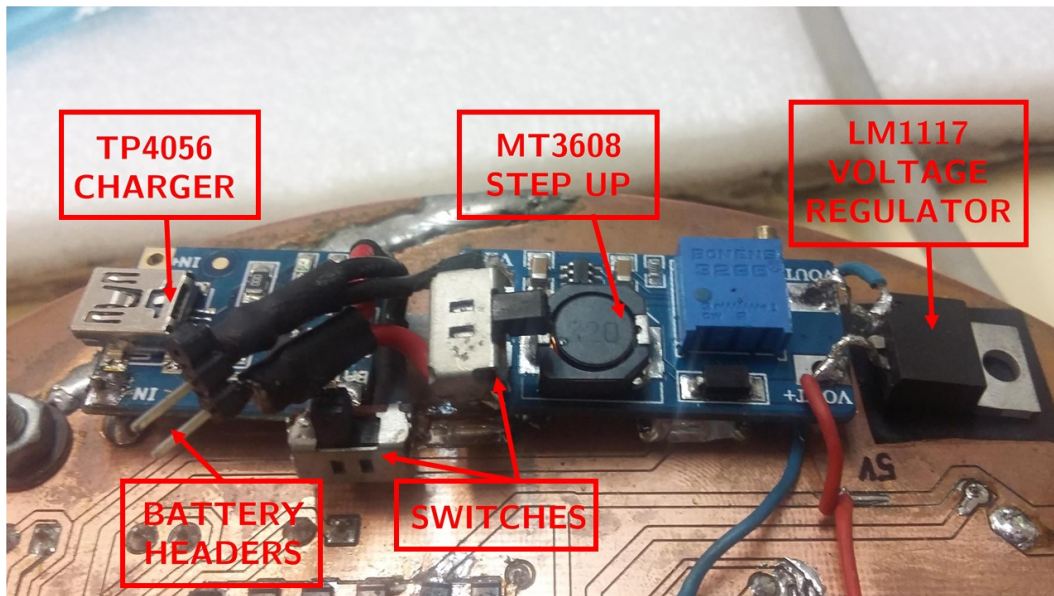


Figure 6.24 – System power soldered to the PCB

## CHAPTER

# 7

# SOFTWARE DESIGN AND IMPLEMENTATION

In section 2.1 we said that we will program the [Testbed](#) as a light detector and this has to be visually appealing for young students. We have all the sensors and elements needed (explained in chapter 4), so we only have to program the [Testbed](#). The light detector should detect the light in the three spacial axis (yaw, pitch a roll). However, our air-bearing only give us a frictionless environment in the base of the [Testbed](#) an the actuators only can turn the [Testbed](#) around Z axis (Yaw axis). Consequently, then we will only have one degree of freedom to detect the light and we will program it in this way. In this chapter we are going to describe in depth how the software design has been done, as well as its implementation.

In section 4.2.1 we said that we will use the microprocesor as an [Arduino](#), so we will program it using [Arduino](#) program language (based on C++) and the [Arduino IDE](#). We will work with [Arduino](#) because it is supported by a big community and there are available a lot of information and a great deal of libraries that simplify the programming. Moreover, [Arduino](#) is open-source and has free license, so we will reduce the costs.

### 7.1 Software general description

The main objective of the software is to implement a light detector. However, we will implement in the [Testbed](#) two different operating mode or “Games” (from now on we will name the operating mode as “games”) to make it visually appealing. These games are:

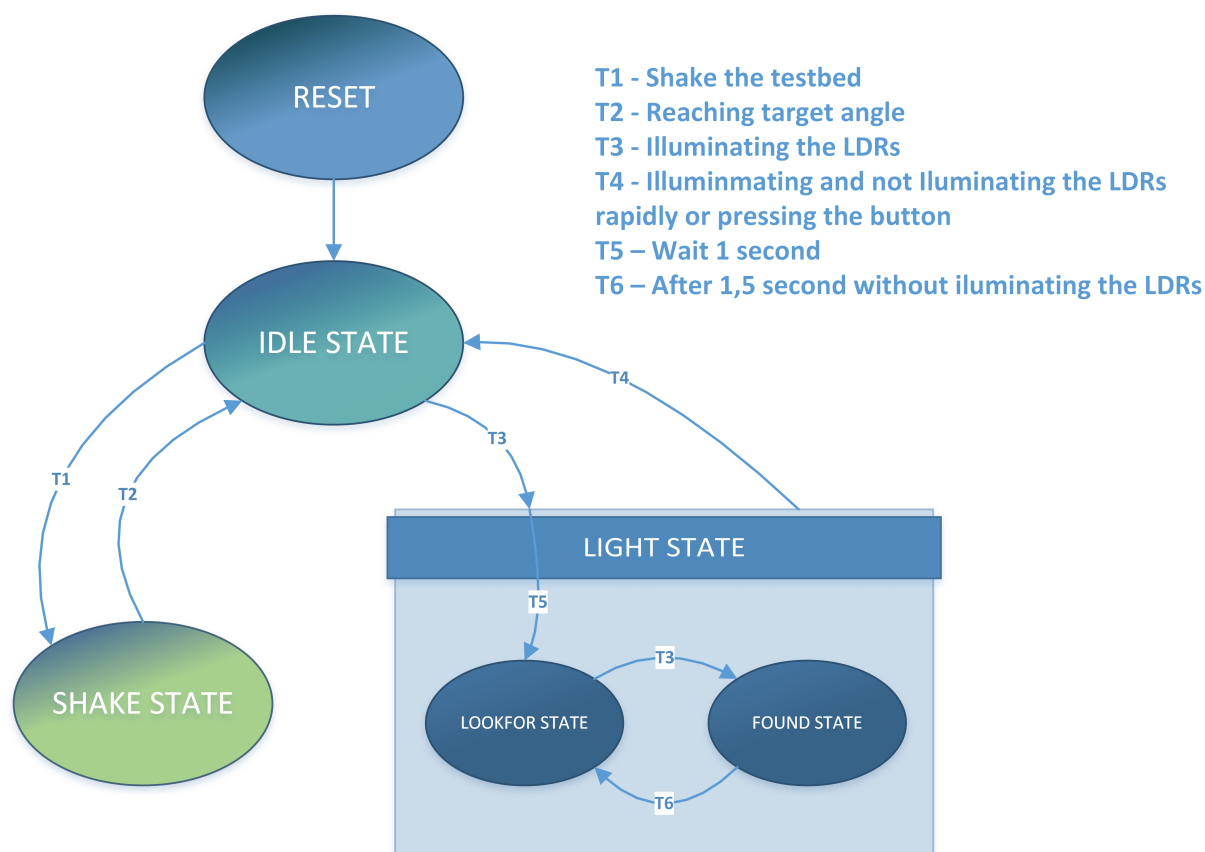
- **Shake game:** When the **Testbed** is idle and it is shaken suddenly, the **Testbed** are going to turn to the original position where it was before being shaken.
- **Light game:** The **Testbed** is going to turn until one of its **LDRs** will be face to face with a light source. If we move this light source, the **Testbed** will turn again looking for the light source.

We will design a **HMI** that allows the communication with the **Testbed** using our hands instead of using buttons or sending commands. This means that the transitions between the two games will be made directly interacting with the **Testbed**. With this we get the **Testbed** that will be striking for young students and moreover we give use to the diffent sensors that are installed on it. However, it will be needed wireless communication between the **Testbed** and the ground station and it will be design and implement in section 7.7

In general terms, it is thought that the **Testbed** will have three different behaviors (two for each game and a third one for when the **Testbed** is idle). So we will define three states, one for each behaviour:

- **IDLE STATE:** When the **Testbed** is switched on or is reseted, it enters directly in the IDLE STATE. In this state the **Testbed** is waiting for you to interact with it and enters in one of the two games.
- **SHAKE STATE:** If the **Testbed** is in IDLE STATE and it is shaken, it enters in SHAKE STATE. In this state, the **Testbed** plays the shake game and tries to turn to the original position. When the original position is reached, the buzzer sounds and, after 5 seconds, the **Testbed** goes back to IDLE STATE
- **LIGHT STATE:** If the **Testbed** is in IDLE STATE and one of its **LDRs** is directly illuminated with a light source (for example, a lantern), it enters in LIGHT STATE. In this state, the **Testbed** plays the light game, which has two substates. After 1 second, the **Testbed** enters in LOOKFOR SUBSTATE and it starts to turn looking for a light source. When one of the **LDRs** is illuminated again, that means that it has found the light source and enters in FOUND STATE and the **Testbed** remains face to face with the light source. Afer 1,5 without illuminating the **LDRs**, the **Testbed** goes back to LOOKFOR STATE. If we want to go back to IDLE STATE, we have to illunate and not illuminate the **LDRs** quickly, similar to do the good bye gesture with the light source. It can be a bit difficult to do the “good bye” gesture while the **Testbed** is turning, so it also is possible to push the button to come back to IDLE STATE

The transitions between these states can be seen graphically in the state diagram of the figure 7.1. Every time the **Testbed** goes to another state, the buzzer sounds to indicate the transition.



**Figure 7.1** – State diagram of the *Testbed*

Although the main objective of the software is to implement the games, there are some subtasks needed to implement the games. These subtasks are:

- To read data from sensors
- To implement a **PID** controller
- To control the actuators
- Communication between the **Testbed** and the ground station

The figure 7.2 shows the block diagram of the software, where we can see the sequence in which each one of the above tasks run. Along this chapter we will describe in depth each one of the main blocks of the diagram (from sections 7.2 to 7.7).

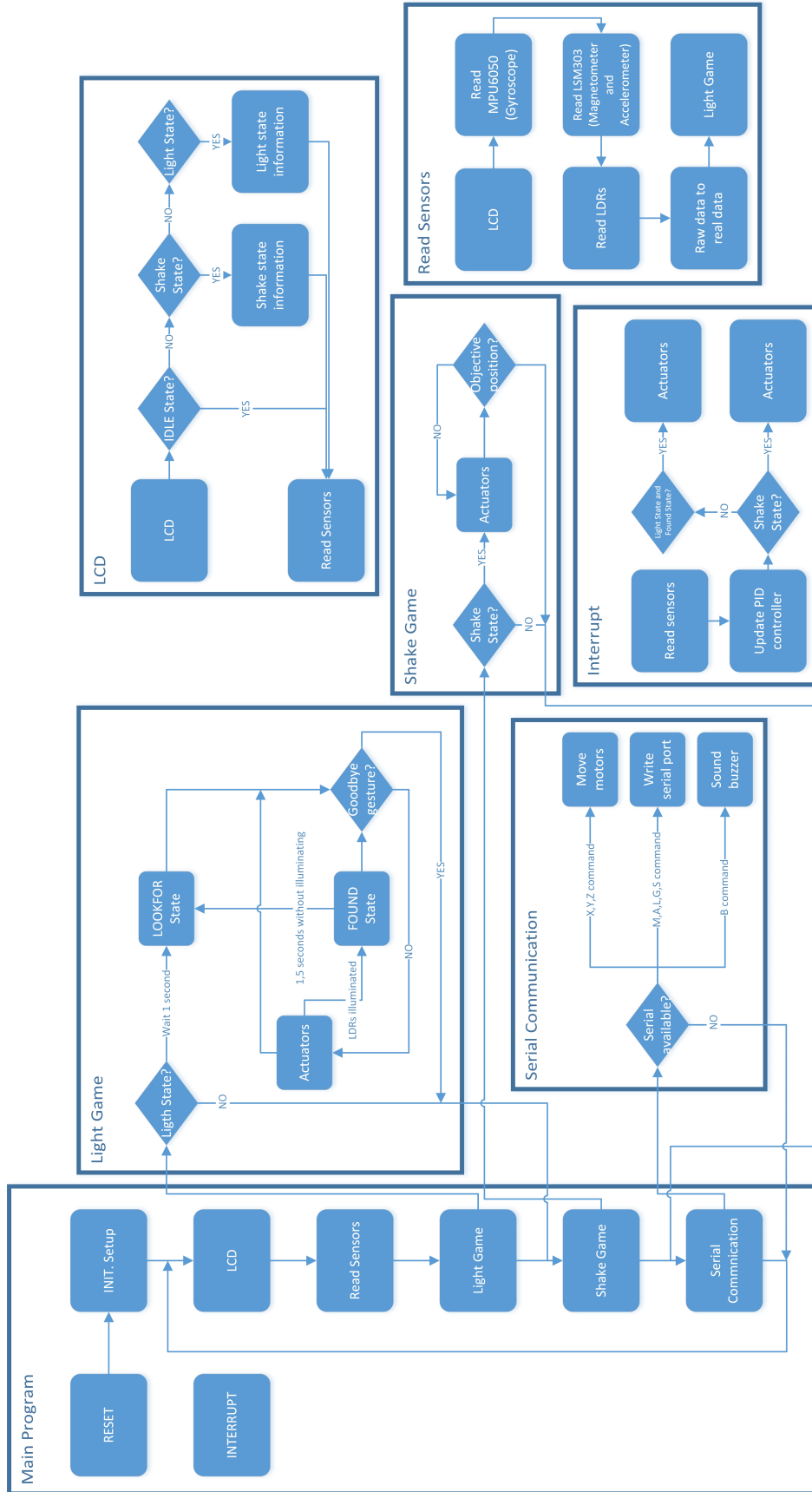


Figure 7.2 – Block diagram of the software

### 7.1.1 Watchdog

We have to remember that, as we said in section 2.2, this software is for aerospace applications, so we have to guarantee that the application will work during extended periods of time. For this reason we will use the watchdog timer.

A watchdog timer is an electronic timer that is used to detect and recover from microprocessor malfunctions. During normal operation, the microprocessor regularly restarts the watchdog timer to prevent it from elapsing, or "timing out". If, due to a hardware fault or program error, the computer fails to restart the watchdog, the timer will elapse and generate a timeout signal. The timeout signal is used to restart the microprocessor [9]. The watchdog timer is a very useful feature for projects that are intended to run for extended periods of time or contain unstable loops of code. If any application the microcontroller freezes, hangs or crashes, the watchdog will time out and will force a reset. It is the same effect that pressing the reset button on the [Arduino](#) board [41].

However, the whatchdog does not work with the [bootloader](#) that the microprocesor had installed so it has been necessary to replace that [bootloader](#) by Optiboot [bootloader](#). This process is explained in appendix A.

### 7.1.2 Real Time Operating System

In section 2.2 we said that one of the requirements for programming the Testbed was to use a [RTOS](#). A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application process data as it comes in, typically without buffering delays. It usually using in integrated applications and they are characterized by special requirements in five general areas [54]:

- Sensitivity
- User Control
- Reliability
- Failure Tolerance
- They are timeshare

We have to remember that our [Testbed](#) will be use to test a future [Cubesat](#) and hence, we are working in a aerospaceal applications, which need lot of reliability requirements. For this reason we decided to programing the [Testbed](#) using a [RTOS](#).

There are a lot of differents [RTOS](#), many of them compatible with [Arduino](#). For our [Testbed](#), we have chosen [FreeRTOS](#) [RTOS](#) because it is free license and behind it has a large

community that supports it. Moreover, there are a lot of documentation, which is very useful to learn to use it. In table 7.1 we can see the main features of the FreeRTOS.

Free RTOS scheduler - preemptive, cooperative and hybrid configuration options, with optional time slicing
The SafeRTOS derivative product provides a high level of confidence in the code integrity
Includes a tickless mode for low power applications
RTOS objects (tasks, queues, semaphores, software timers, mutexes and event groups) can be created using either dynamically or statically allocated RAM
Tiny footprint
Official support for >30 embedded system architectures (including ATmega328)
Designed to be small, simple and easy to use. Typically a RTOS kernel binary image will be in the region of 4K to 9K bytes
Very portable source code structure, predominantly written in C
Supports both real time tasks and co-routines
Direct to task notifications, queues, binary semaphores, counting semaphores, recursive semaphores and mutexes for communication and synchronisation between tasks, or between real time tasks and interrupts
Innovative event group (or event flag) implementation
Mutexes with priority inheritance
Efficient software timers
Powerful execution trace functionality
Stack overflow detection options
Pre-configured RTOS demo applications for selected single board computers allowing 'out of the box' operation and fast learning curve
Free monitored forum support, or optional commercial support and licensing
No software restriction on the number of real time tasks that can be created
No software restriction on the number of task priorities that can be used
No restrictions imposed on task priority assignment - more than one real time task can be assigned the same priority
Free development tools for many supported architectures
Free embedded software source code
Royalty free
Cross development from a standard Windows host

**Table 7.1** – *FreeRTOS main features [12]*

FreeRTOS uses very little memory. In table 7.2 we can see how much memory it exactly uses. For example, a simple FreeRTOS program that switch on and switch off a LED according with a signal takes 8.426 bytes of the microprocessor memory (26% of the ATmega328 memory). This means that we cannot use FreeRTOS unless we change the microprocessor for another one with more flash memory.

Item	Bytes Used
Scheduler Itself smaller data types)	236 bytes (can easily be reduced by using
For each queue you create	76 bytes + queue storage area
For each task you create characters for the task name) + the task stack size	64 bytes (includes 4

**Table 7.2** – *FreeRTOS memory usage [13]*

However, although the FreeRTOS uses low memory, the sun detector program takes 28.192 bytes bytes (87% of the ATmega328 memory) without using FreeRTOS. Consequently, the ATmega328's memory is not enough if we would include the FreeRTOS code.

## 7.2 Initial setup

[Arduino sketches](#)<sup>1</sup> always have the same structure. First is the `setup()` block and it is ran once. Then is the `loop()` block and it runs constantly in a looped way. The initial setup will be inside the `setup()` block and the other tasks will be inside the `loop()` block.

When the [Testbed](#) is switch on or come from a reset, the first task to run is the initial setup. In this task, the software variables are defined, as well as the input and output pins of the microprocessor. Moreover, is here where we have to set up the  $I^2C$  communication establishing the  $I^2C$  address for MPU6050 and LSM303 devices, as well as the LCD screen. These devices have to be initialized too. As we said in section 4.2.4.1.1, the MPU6050's [DMP](#) has to be calibrated. In the initial setup is where we set the [DMP](#) offsets.

In figure 7.1 we can see that the first (and only) state in which the [Testbed](#) can stay after the reset (or after being switched on) is the *Idle State*. So in the initial setup we have to establish the current state of the [Testbed](#) as idle state.

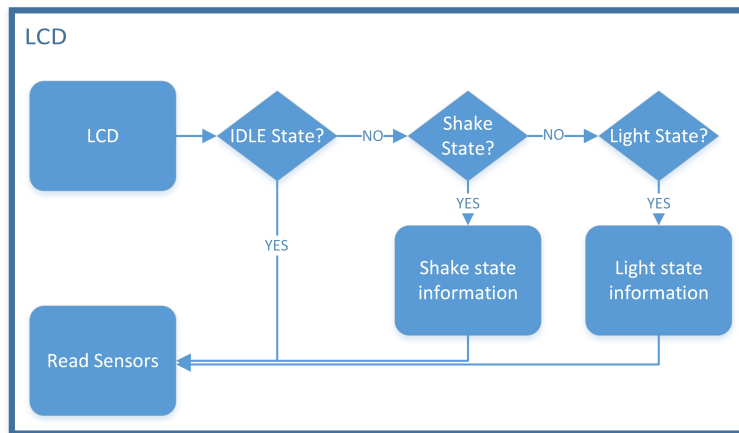
There will be wireless communication between the [Testbed](#) and the ground station using [XBee](#) modules. This module sends data by serial port at 9600 bauds, so we have to enable the serial port communication in the initial setup at 9600 baud rate. On the other hand, in figure 7.2 we can see that in the main program block appears an interrupt block. This interrupt is defined in the initial setup too. In section 4.4 we will explain in depth what the interrupt does and how often runs.

Finally, in section 7.1.1 we said that it was necessary to include a watchdog that avoid that reset the program if it freezes. The watchdog is set in the initial setup and, in our case, the watchdog will reset the program after 8 seconds frozen.

<sup>1</sup>Name given to the source code opened and compiled with the [Arduino IDE](#)



### 7.3 LCD screen



**Figure 7.3** – Block diagram of the LCD task

The LCD task is responsible for displaying information on LCD screen. The information shown depends on the current state in which the program is. As we can see in the block diagram of figure 7.3, first the program checks if the Testbed is in *idle state*. In this case, the program goes to the next task (Read Sensor) and only displayed by the screen the current state (*idle State*) without any additional information. If not, the program checks if the Testbed is in *shake state* or in *light state* (in this order). If the current state is the *shake state*, the LCD displays “Shake Game” in the first line and in the second line it is displayed “O:” + the target angle and “E:” + the error angle (difference between the target angle and the current angle) as we can see in figure 7.4. On the other hand, if the current state is the *Light State*, we have to differentiate between its two substates (*Lookfor State* and *Found State*). If the program is in *Lookfor State*, the LCD displays “Light State” in the first line and “Finding Sun” in the second line (we will suppose that the light source is the sun) as we can see in figure 7.5. Finally, if the program is in *Found State*, the LCD displays “Found” in the first line and in the second line it is displayed “O:” + the target angle and “E:” + the error angle (difference between the target angle and the current angle) in the same way that in *Shake State*, as we can see in figure 7.6. In summary, table 7.3 shows the information displayed in each state.



**Figure 7.4** – *Shake game screen*



**Figure 7.5** – *Finding state screen*

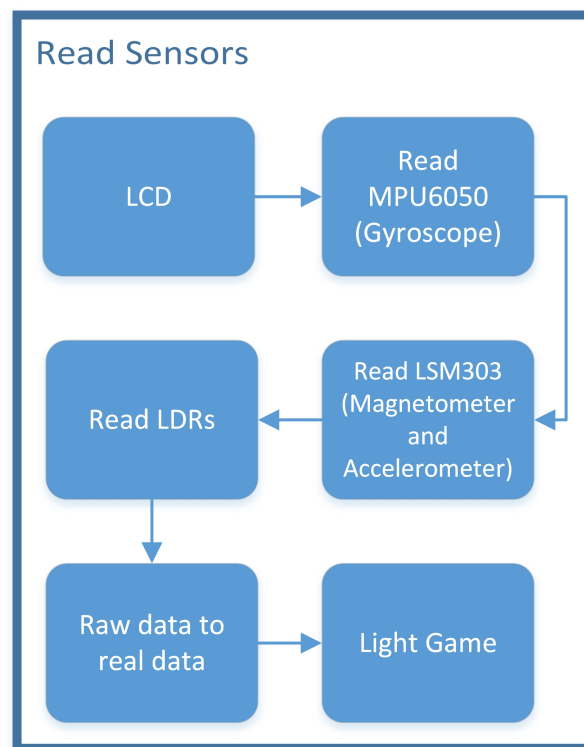


**Figure 7.6** – *Found state screen*

State	Information shown
Idle State	Current State
Shake State	Current State/Target angle/Distance to target angle
Light State: Lookfor State	Current State/ “Finding Sun”
Light State: Found State	“Found”/ Target angle/ Distance to target angle

**Table 7.3** – Information displayed for each state

## 7.4 Reading Sensors



**Figure 7.7** – Block diagram of the Read Sensor task

# 7

The reading sensors are one of the most important tasks because the actuators depend on these readings. In figure 7.2 we can see that the reading sensors task to run sequentially in the main loop between LCD and Light Game tasks. However, the reading sensors task will also run inside the interrupt, as it will be described in section 7.8.

Figure 7.7 shows the block diagram of the reading sensor task. As we can see, first we read the MPU6050 device, the the LSM303 device and finally, the LDRs. For MPU6050 and LSM303 devices, we have used the code available in I<sup>2</sup>Cdevlib library [34] to read the raw data and, in MPU6050 case, we have used the DMP to convert the giroscope raw data in

yaw, pitch and roll readings (this code is also available in I<sup>2</sup>Cdevlib [34]). Finally, the LDRs readings are very simple because we only have to read the value of the analogical pins where the LDR are connected.

We have to say that both MPU6050 and LSM303 devices have an accelerometer so we can read this information from both devices. We have chosen to read the accelerometer from LSM303 because the code is simpler. Moreover, the MPU6050 has to do an extra work to convert the raw data in yaw, pitch and roll angles, so LSM303 will be less loaded.

Once we have read the devices, we have to convert the raw data in real data. This process was explained in section 4.2.4. For MPU6050, although the DMP has converted the raw data in yaw, pitch a roll data, we are going to use equations from 4.2.1 to 4.2.3 to express the rotating of each axis in degrees from -180 to 180. For LSM303 we only have to include in the code the equations from 4.2.4 to 4.2.9 to correct the scale factor for both accelerometer and magnetometer. Finally, for LDRs reading, we include the code to map the reading valuee as we explained in section 4.2.4.3.

## 7.5 Games

The “Light Game” and “Shake Game” tasks are responsible for enter the program in *Light State* and *Shake State* according with the sensor readings and the interaction between the user and the Testbed. In this section we will describe each one of these tasks in depth in sections 7.5.1 and 7.5.2 respectively.

### 7.5.1 Light Game

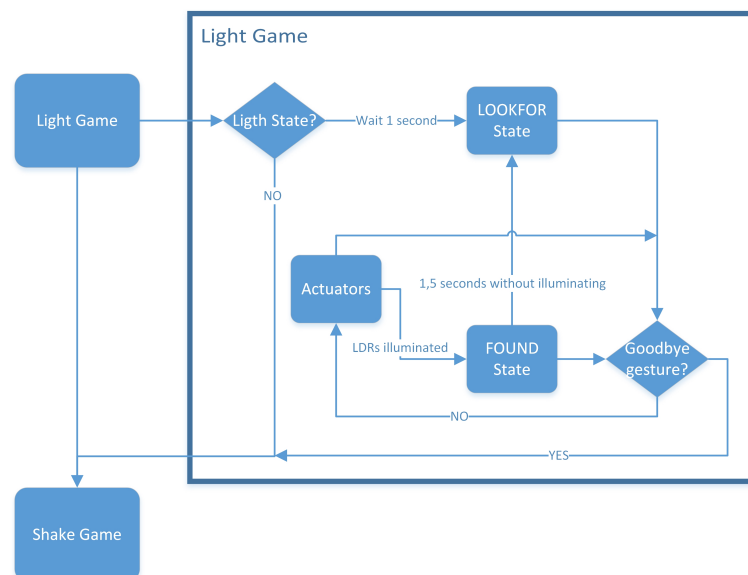
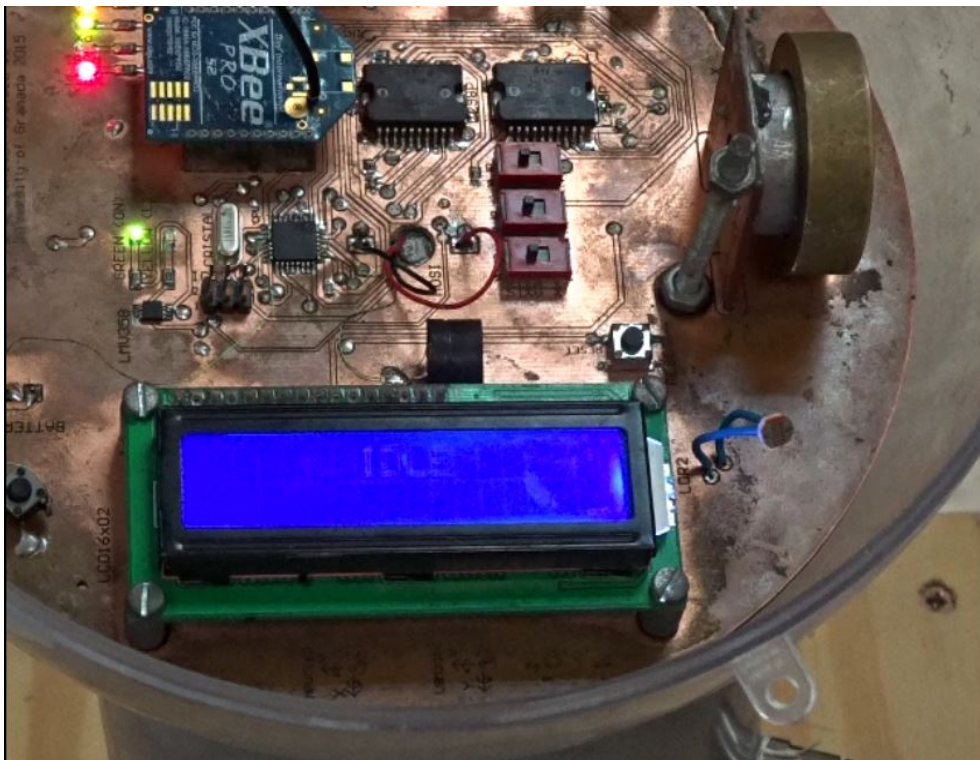


Figure 7.8 – Block diagram of the light game

In figure 7.8 we can see the block diagram of the light game task. As we said in the light game description in section 7.1, the program enter in the *Light State* when one of the LDRs is directly illuminated by a light source so the first thing that the light game task does is to check it. We have defined an “illumination threshold”. That means that if the read value from the LDR is higher than this “illumination threshold”, then the LDR is directly illuminated and the system goes into the *Light State*. If not, the LDR is not directly illuminated and the program goes to the next task (*Shake Game*).

Once the program has gone into *Light State*, it waits for 1 second and the program goes into *Lookfor State*. It is necessary to wait in order to give time to the user to remove the light source and to avoid that the program goes into *Found State* directly. Then the Z axis motor will start to turn (not too fast) and consequently, the *Testbed* will turn too. When the program is in *Lookfor State*, it is possible to came back to *Idle State* doing the “goodbye gesture” which we described in section 7.1. This is to illuminate one the LDR, remove the light source and quicky illuminate it again, doing all this process in a very short time (500 ms). This gesture can be seen more clearly in video 7.1. If we do not do the “goodbye gesture”, the *Testbed* will turn until one of the LDRs will be directly illumintad again, and the program will go into *Found State*.



Video 7.1 – “GoodBye gesture”

If there is any problem to see the video correctly, see [https://www.youtube.com/watch?v=zFYr3sEshZc&feature=youtu.be&ab\\_channel=GranaSATTeam](https://www.youtube.com/watch?v=zFYr3sEshZc&feature=youtu.be&ab_channel=GranaSATTeam).

As we said at the beginning of the chapter (7.1), doing the “goodbye gesture” can be a bit difficult so it is possible to push the button to come back to *Idle State*.

The objective of the *Found State* is to put face to face the LDRs and the light source. When the program goes into this state, the Z axis motor brakes sharply during one second in order to stop the *Testbed* and then, the Z axis motor actuator are going to turn according to the PID controller, which will be explained in depth in section 4.4. However, if the LDRs are not illuminated during 1,5 seconds this means that the light source has been removed and the program comes back to *Lookfor State* again. When the program goes into the *Found State*, its behavior is very similar to the (*Shake Game*)’s behaviour. The difference is that in this case the target angle is the angle where the illuminated LDR is. To know this angle, we have to take into account the position of the LDRs in the PCB. Figure 7.9 shows the angle of the two LDRs respect to the  $0^\circ$  angle taken by the gyroscope.

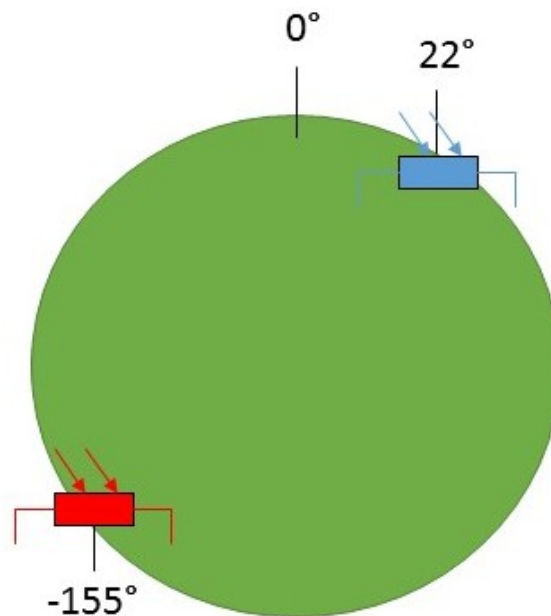
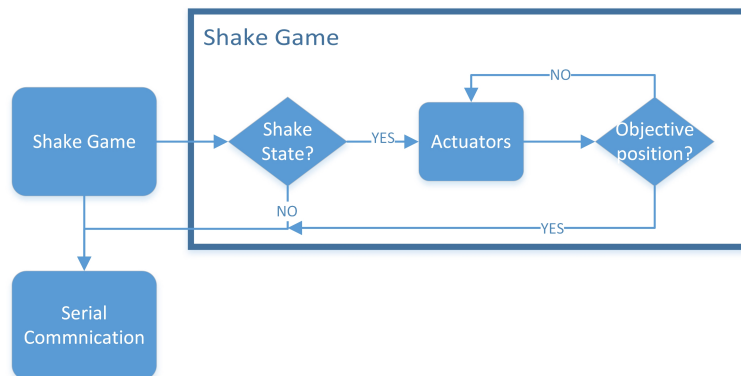


Figure 7.9 – Position of the LDR respect to  $0^\circ$

## 7.5.2 Shake Game



**Figure 7.10** – Block diagram of the shake game

In figure 7.10 we can see the block diagram of the shake game task, which works similar to the light game task but in a simpler way because there is not any substate. The program goes into *Shake State* when the *Testbed* is shaken. To detect this shake, we are going to use the accelerometer. It is logical to think that when an object is shaken suddenly, its acceleration is increased. So, in the same way that in 7.5.1, we will define a “acceleration threshold”. If the acceleration read by the sensor in one of the three axes is higher than the “acceleration threshold”, then the *Testbed* are being shaken and the program goes into *Shake State*. If not, the program goes to the next task (Serial Communication).

We have to take into account that the “acceleration threshold” must be different for Z axis because it is affected by the gravity, so the acceleration readings in this axis will be 9.8 higher than in the other two axes. Once the program has gone into *Shake State*, it automatically takes the current Z angle as target angle. Then the idea is that the user places the *Testbed* in and different angle above the *air-bearing* and the *Testbed* will turn until reach the target Z angle.

After checking that the program is in *Shake State*, the Z axis motor will turn according the *PID* controller (section 4.4). When the *Testbed* reaches the target angle, the buzzer sounds and if the *Testbed* is in the target angle during 5 second, the goal is achieved and the program leave the *Shake State* and comes back to the *Idle State*.

# 7

## 7.6 PID controller implementation

In section 4.4 we explained a brief introduction about *PID* controllers. It is very important a good implementation because this controller is responsible to move the actuators in order to the *Testbed* reaches the target angle. As we said in section 4.4, to tuning the *PID* controller using the Ziegler-Nichols technique we have to calculate the parameters of the table 4.25, but first we have to calculate the step response of the system.

### 7.6.1 Step response

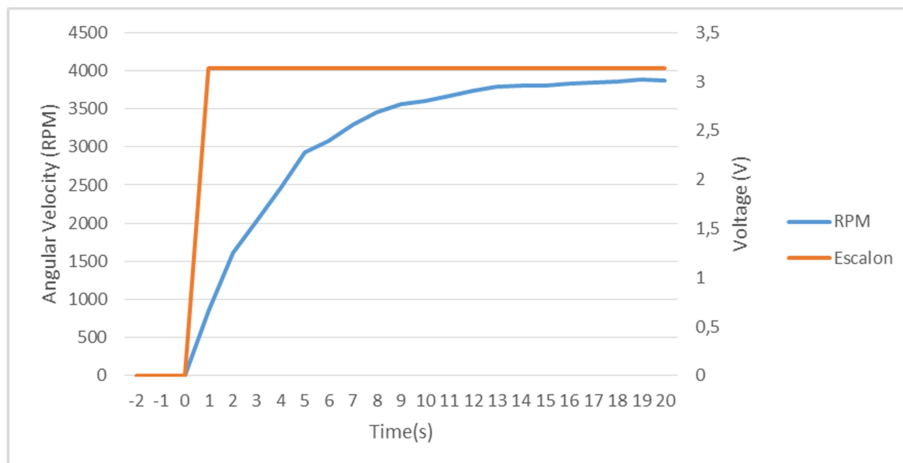
Victor Burgos in his bachelor thesis said that the angular velocity step response will be the decisive data to calculate the PID parameters [21]. Then, we are going to measure the angular velocity in RPM of each reaction wheel each second, since we switch on the motors until 20 seconds later. We are going to take three measurements and then we will calculate the average in order to have a more accurate result. The voltage between the motor terminal is 3,14 volts when we switch on it.

In table 7.4 we can see the values measurements for X axis reaction wheel, while in figure 7.11 we can see a graphic representation of the average RPM and the step response.

Time(s)	Measure1(RPM)	Measure2(RPM)	Measure3(RPM)	Average(RPM)
-2	0	0	0	0
-1	0	0	0	0
0	0	0	0	0
1	792	793,7	948	844,57
2	1569	1607	1660	1612
3	2132	2088	1851	2023,67
4	2571	2340	2488	2466,33
5	2853	2949	2973	2925
6	3021	3131	3066	3072,67
7	3263	3252	3366	3292,67
8	3466	3413	3493	3457,33
9	3521	3563	3589	3557,67
10	3586	3573	3668	3609
11	3649	3677	3690	3672
12	3716	3737	3773	3742
13	3763	3796	3820	3793
14	3785	3802	3843	3810
15	3790	3776	3852	3806
16	3795	3830	3862	3829
17	3797	3850	3872	3839,67
18	3803	3871	3888	3854
19	3811	3889	3961	3887
20	3813	3897	3898	3869,33

Table 7.4 – Step response for X axis motor



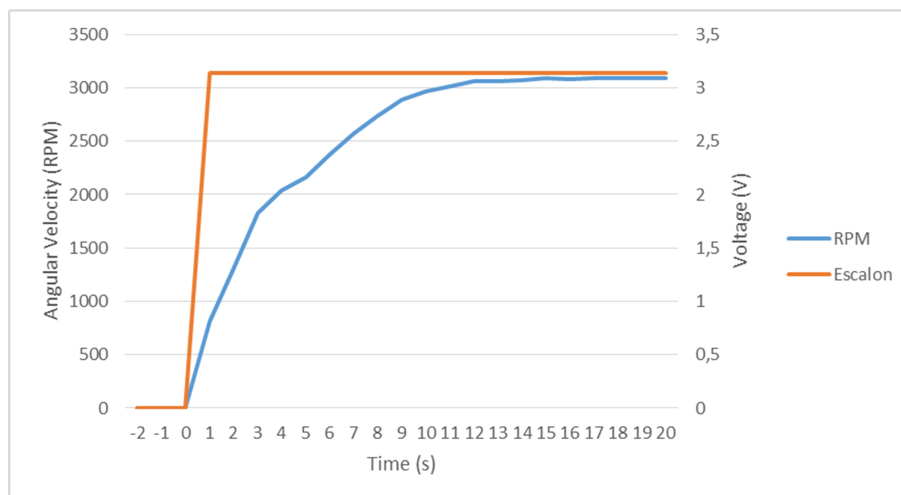


**Figure 7.11** – Step response for X axis motor

In the same way, tables 7.5 and 7.6 show the values measurements for Y and Z axis reaction wheels respectively, and figures 7.12 and 7.13 show the graphical representation of their step responses.

Time(s)	Measure1(RPM)	Measure2(RPM)	Measure3(RPM)	Average(RPM)
-2	0	0	0	0
-1	0	0	0	0
0	0	0	0	0
1	737,6	888,9	803	809,833
2	1334	1292	1274	1300
3	1792	1851	1838	1827
4	1956	2040	2111	2035,67
5	2102	2223	2170	2165
6	2323	2447	2344	2371,33
7	2574	2588	2563	2575
8	2656	2735	2817	2736
9	2885	2868	2924	2892,33
10	2948	2921	3026	2965
11	2966	2985	3079	3010
12	3039	3047	3094	3060
13	3045	3049	3093	3062,33
14	3050	3086	3082	3072,67
15	3065	3112	3105	3094
16	3074	3088	3079	3080,33
17	3079	3113	3093	3095
18	3063	3086	3115	3088
19	3071	3091	3097	3086,33
20	3081	3104	3081	3088,67

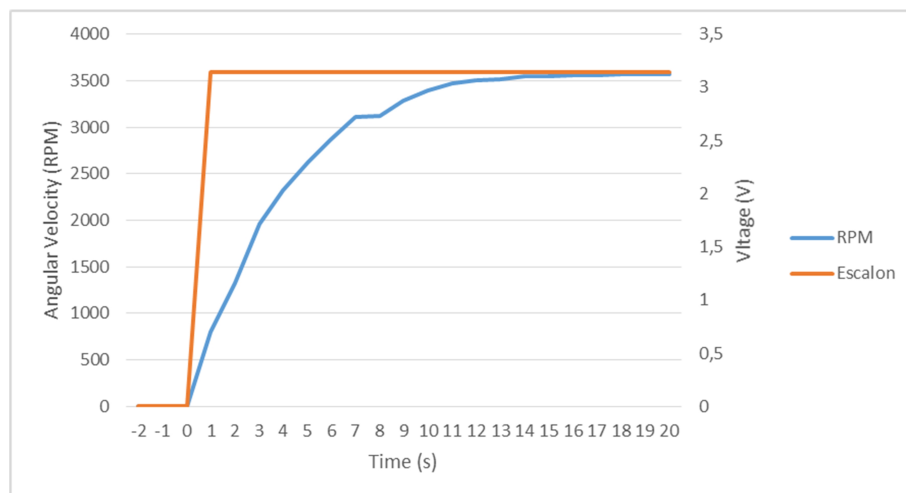
**Table 7.5** – Step response for Y axis motor



**Figure 7.12** – Step response for Y axis motor

Time(s)	Measure1(RPM)	Measure2(RPM)	Measure3(RPM)	Average(RPM)
-2	0	0	0	0
-1	0	0	0	0
0	0	0	0	0
1	877	823	714	804,67
2	1312	1322	1333	1322,33
3	1982	1947	1951	1960
4	2300	2319	2339	2319,33
5	2577	2658	2623	2619,33
6	2849	2878	2882	2869,67
7	3118	3099	3113	3110
8	3125	3124	3103	3117,33
9	3220	3294	3346	3286,67
10	3421	3358	3409	3396
11	3440	3473	3484	3465,67
12	3506	3492	3502	3500
13	3505	3514	3521	3513,33
14	3538	3541	3547	3542
15	3551	3550	3547	3549,33
16	3562	3560	3558	3560
17	3554	3561	3570	3561,67
18	3563	3574	3560	3565,67
19	3562	3576	3571	3569,67
20	3565	3571	3565	3567

**Table 7.6** – Step response for Z axis motor



**Figure 7.13** – Step response for Z axis motor

## 7.6.2 Parameter calculation

As we said in 4.4, we need  $T_1$  (rise time) and  $T_D$  (response times) times to calculate the parameters of the table 4.25. We can calculate  $T_1$  easily thanks to the graphical representation of the step response. However,  $T_D$  is more difficult to calculate because it have miliseconds scale and it is not appreciable. In section 7.6.2.2 we will explain how we have measured this time.

### 7.6.2.1 Rise time

According with [14],  $T_1$  or rise time is calculated from the point at which the tangent line intersects the initial value of the system to the point where the tangent line reaches the final value of the system. We can see this process done for each reaction wheel in figures 7.14, 7.15 and 7.16.

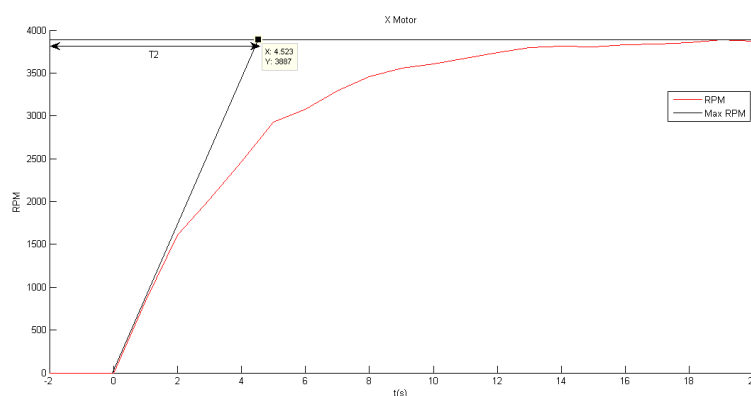


Figure 7.14 – Rise time for X axis motor

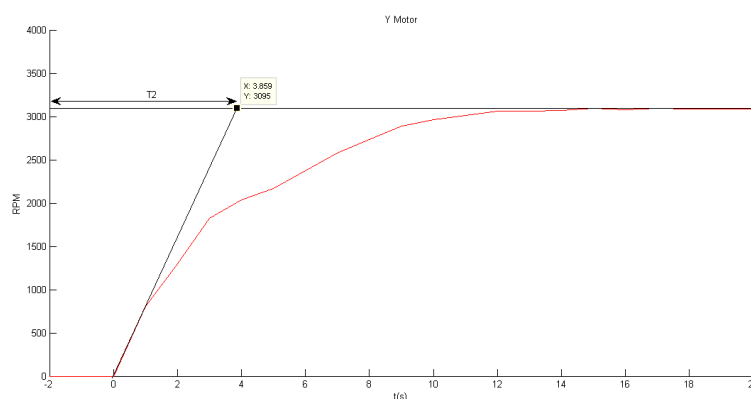
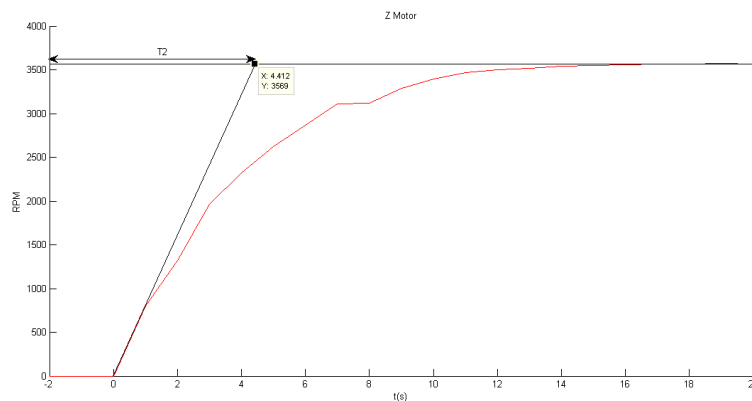


Figure 7.15 – Rise time for Y axis motor



**Figure 7.16** – Rise time for Z axis motor

Table 7.7 summarizes the rise time calculated for each reaction wheel.

X reaction wheel	4,523 seconds
Y reaction wheel	3,859 seconds
Z reaction wheel	4,412 seconds

**Table 7.7** – Rise time for each motor

### 7.6.2.2 Response time

The response time is the time it takes the system to begin to response. In our case, it is the time elapsed between the microprocesor give the order and the motor begin to turn. Unlike the rise time, the response time is very much difficult to measure. In other systems, this time is possible to be measured only using a chronometer but, in our case, it is impossible. On the basis that the motors begin to turn at maximum power when the current through the motor is maximum, we will measure the time elapsed between we give the order to turn and the maximum current point.

The problem is that measure the current through the motor when it is welded to the PCB is not a trivial task becuase the current must be serial measured. However, the voltage between the motor terminals can be easily measured, for example, using an oscillosope. Knowing the voltage and the resistance we can know the current thanks to the Ohm law (equation 7.6.1).

$$I = \frac{V}{R} \quad (7.6.1)$$

But we still have to know how much is the resistance. To be ensure exactly how much is the resistance, we are going to shortcircuit the track that links the motor with the microprocesor.

Then, we are going to weld a  $1\ \Omega$  resistance (figure 7.17) in the track, doing a bridge over the short. Now we can measure the voltage in both resistor terminals (figure 7.18). Due to the resistance is  $1\ \Omega$ , the current will be equal to the voltage (equation 7.6.1).

This is a difficult process and the tracks are not easily accessible for the three motors. For these reasons and because the three motors are very similar, we will suppose that the response time is equal for the three motors and we will only do this process once.



Figure 7.17 –  $1\ \Omega$  resistance



Figure 7.18 –  $1\ \Omega$  resistance solded to the *PCB*

Figure 7.19 shows a capture of the oscilloscope screen. The yellow signal is the voltage measured after the resistance and the green signal is the voltage measured before the

resistance. The purple signal is the difference between the green signal and the yellow signal or, in other words, the voltage drops in the resistance. As the resistance is  $1\ \Omega$ , the purple signal is equal to the current through the resistance and through the motor.

Figure 7.20 shows an expansion of the purple signal. In this figure we can see the rise of the signal. With te cursors we have measured this time and, as we can see inside the red rectangle, we have obtained a response time of 1.12 ms. Once we know this data, we have to remove the resistance and bridge the shortcircuit.

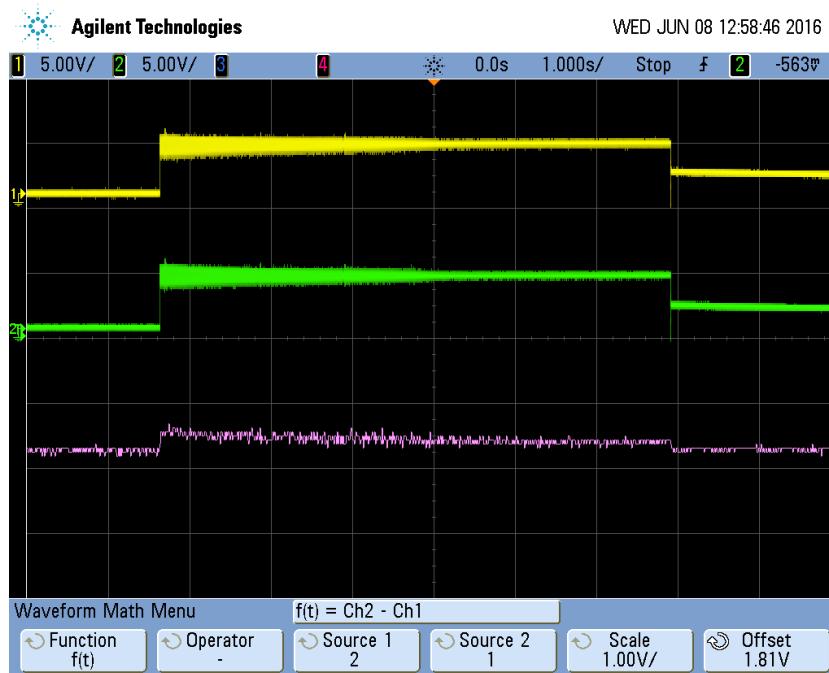


Figure 7.19 – Capture of the oscilloscope screen

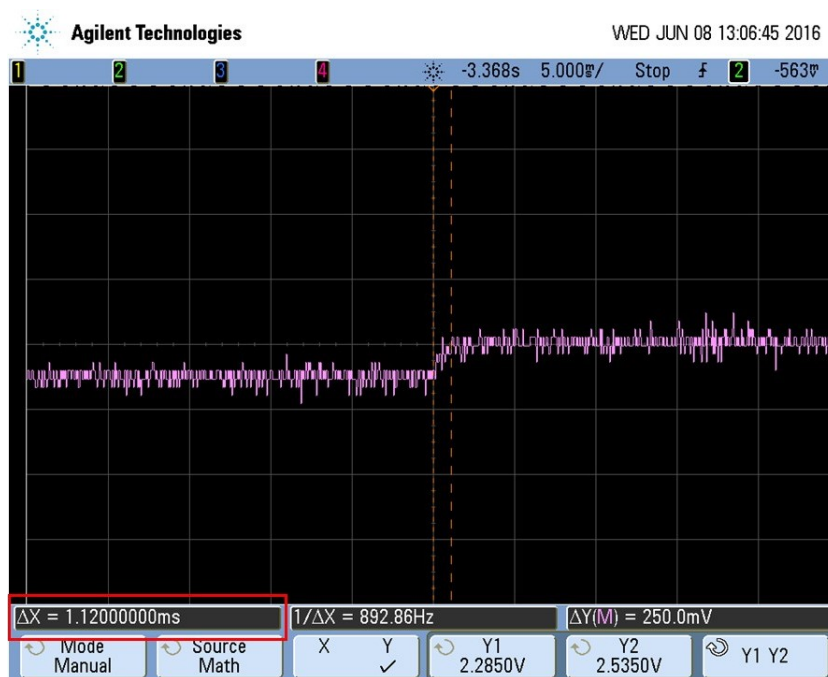


Figure 7.20 – Extended capture of the oscilloscope screen

### 7.6.2.3 Final parameter calculation

Now we have all the needed data to calculate the parameters of the table 4.25. The rise time  $T_1$  and the response time  $T_D$  have been calculated in sections 7.6.2.1 and 7.6.2.2 respectively. The variation of the step  $K_s$  and the variation of the system response can be obtained from the step response (7.6.1). Table 7.8 summarizes these data for each axis. Table 7.9, 7.10 and 7.11 show the result for X, Y and Z axes respectively.

Axis	$T_1$	$T_D$	$K_s$	$K$
X	4,523 s	1.12 ms	3.14 V	3869,33 RPM
Y	3,859 s	1.12 ms	3.14 V	3088,67 RPM
Z	4,412 s	1.12 ms	3.14 V	3567 RPM

Table 7.8 – Needed data to calculate the Zieger Nichols parameters

Controller	$K_p$	$T_i$	$T_d$
Proportional (P)	3,28		
Proportional + integral (P+I)	2,95	0,003696	
Proportional + integral + Derivative (P+I+D)	3,93	0,00224	0,00056

Table 7.9 – Results of Zieger-Nichols parameters for X axis



Controller	$K_p$	$T_i$	$T_d$
Proportional (P)	3,50		
Proportional + integral (P+I)	3,15	0,003696	
Proportional + integral + Derivative (P+I+D)	4,20	0,00224	0,00056

**Table 7.10** – Results of Zieger-Nichols parameters for Y axis

Controller	$K_p$	$T_i$	$T_d$
Proportional (P)	3,47		
Proportional + integral (P+I)	3,12	0,003696	
Proportional + integral + Derivative (P+I+D)	4,17	0,00224	0,00056

**Table 7.11** – Results of Zieger-Nichols parameters for Z axis

The tables above show the results in function of  $T_i$  and  $T_d$ . However we are going to use the  $K_i$  and  $K_d$  constants so we will use the equations 4.4.4 and 4.4.5 to show the results in fuction of  $K_i$  and  $K_d$  in tables 7.12, 7.13 and 7.14.

Controller	$K_p$	$K_i$	$K_d$
Proportional (P)	3,28		
Proportional + integral (P+I)	2,95	798,02	
Proportional + integral + Derivative (P+I+D)	3,93	1755,64	0,0022

**Table 7.12** – Results of Zieger-Nichols parameters for X axis

Controller	$K_p$	$K_i$	$K_d$
Proportional (P)	3,50		
Proportional + integral (P+I)	3,15	852,95	
Proportional + integral + Derivative (P+I+D)	4,20	1876,50	0,0024

**Table 7.13** – Results of Zieger-Nichols parameters for Y axis

Controller	$K_p$	$K_i$	$K_d$
Proportional (P)	3,47		
Proportional + integral (P+I)	3,12	844,41	
Proportional + integral + Derivative (P+I+D)	4,17	1857,70	0,0023

**Table 7.14** – Results of Zieger-Nichols parameters for Z axis

### 7.6.3 Choosing a sampling interval

If we see the equation 4.4.2, we can observe that we have a sample time  $T$ . This time means the period in which the PID controller updates the speed of the reaction wheels. According to [35], the simplest election is to sample as fast as possible. However, unnecessarily fast sampling is a waste of resources. Moreover, [35] said that if the system has the Ziegler-Nichols open-loop response as given by equation 4.4.2, then we have to choose the sampling time as  $T < \frac{T_1}{10}$ .

On the other hand, is illogical to choose a sample time lower than the time it takes to run the PID controller program. We are going to measure how much is this time to know the limits of our choice. There are two main functions responsible for the PID controller, one function responsible for reading the sensors and another one responsible for doing the calculations. To measure how much time is taken to run both functions, we are going to write a logical “1” in the beginning of each function and then we will write a logical “0” at the end in a specific pin of the microprocessor. Using an oscilloscope we can measure the time seeing the distance in the X axis between the peaks.

First, we have measured the time for the read sensors function. Figure 7.21 shows the process described above. As we can see in the red rectangle, the time between the peaks is 10 ms.

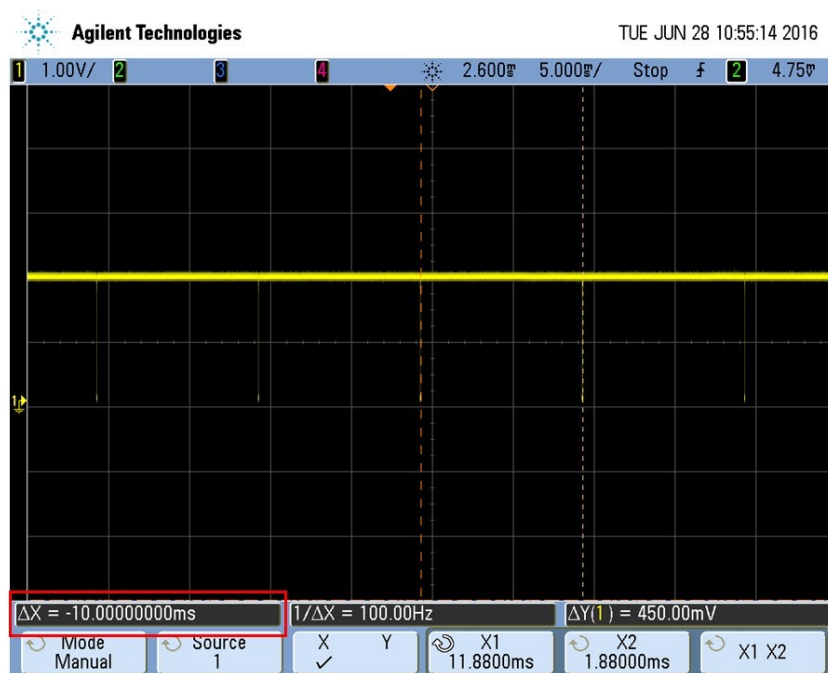


Figure 7.21 – Time taken to read the sensors

Now we are going to do the same process for the calculation function. Figure 7.22 shows the time while the pin is up. As we can see in the red rectangle, this time is 46 ms.



Figure 7.22 – Time taken to do the PID calculations

Consequently, the total time that the PID controller takes to run is the sum of the two parcial times. The PID controller takes 56 ms to run.

The sample time must be higher than 56 ms and lower than  $T < \frac{T_1}{10}$ . Table 7.15 shows a summarize of the minimum and maximum sample time for each axis.

Axis	Minimum	Maximum
X	56 ms	452 ms
Y	56 ms	386 ms
Z	56 ms	441 ms

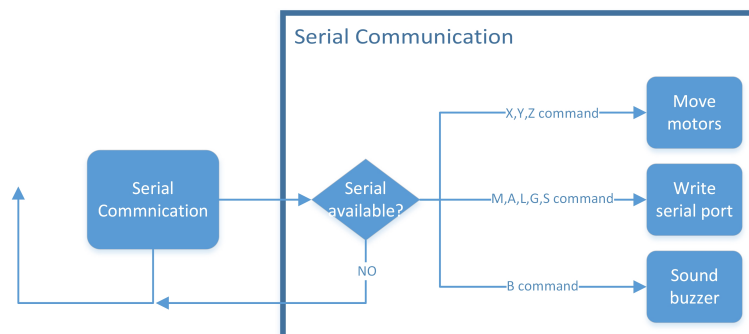
Table 7.15 – Minimum and maximum sample time for each axis

7

Moreover, we have to take into account that if we choose the minimum sample time, then the program would be persistently updating the PID controller and it did not give time to others software functions to run. For this reason, the sample time has to be significantly larger than the minimum sample time.

Taking all this questions into account, we have choosen a sample time of 150 ms.

## 7.7 Communication software



**Figure 7.23** – Block diagram of the serial communication task

If we see the program block diagram (figure 7.2), the next task after the shake game in the main loop is the serial communication. This task is responsible of the communication between the [Testbed](#) and the ground station. Although we have only used this characteristic for testing, in the future it could be improved to use it for monitoring the [Testbed](#) since the ground station.

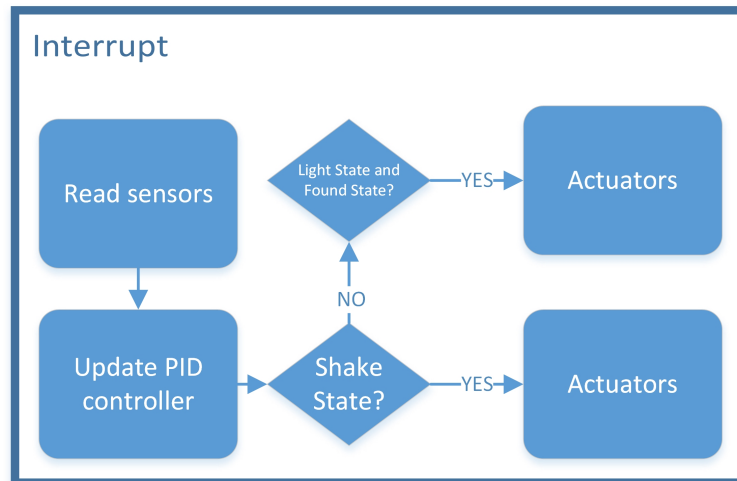
We have desinged a very simple protocol. As we can see in the figure 7.23, if the program have some command character available in the serial port, it makes an action based on the character. If not, the program goes to the next task (as it is the last task in the loop, the loop start again since [LCD](#) task). We can see that there are three different types of commands. A first type of command return information about the sensors or about the [Testbed](#) state to the ground station. The others two types makes the actuators work. The difference between them is their command syntax. Table 7.16 shows all the commands available with a short description.

Command	Description
M	Returns magnetometer data
A	Returns accelerometer data
L	Returns light sensors data
G	Returns gyroscope data
B	Sounds the buzzer
S	Return the current <a href="#">Testbed</a> state
X <direction> <speed>	Turn X axis motor
Y <direction> <speed>	Turn Y axis motor
Z <direction> <speed>	Turn Z axis motor

**Table 7.16** – Commands to communicate with the [Testbed](#)

As we can see, all the commands are composed by only one character except the commands that make the actuators to move. In this case, we have to specify the direction (“*backward*” or “*forward*”) and the velocity with a number between 0 and 255 (for more information about the relationship between the **PWM** values and the **RPM** obtained, see the section 4.2.5).

## 7.8 Interrupt



**Figure 7.24** – Block diagram of the serial interrupt task

The last block that we have to explain is the interrupt block. As its name suggests, this block implements a programed interrupt or timer. This timer allows tu run a function when it has elapsed a certain time, previously programed. In section 7.6.3, we said that the **PID** controller needed a sample time and we chose an apropiated one (150 ms). We will use the interrupt timer to run and update the **PID** controller accurately every 150 ms. In this way, when the interrupt is triggered, the main program loops stop to run and the **PID** controller is updated.

As we can see in the figure 7.24, the first task to run when the interrupt is triggered is to read sensors because it is a necessary step before update the **PID** controller. Then, it is checked what is the current state of the **Testbed**. If the program is in *Shake State* or in *Light State (Found Substate)*, the actuators turn according to the **PID** controller. If not, the interrupt do not do any action.

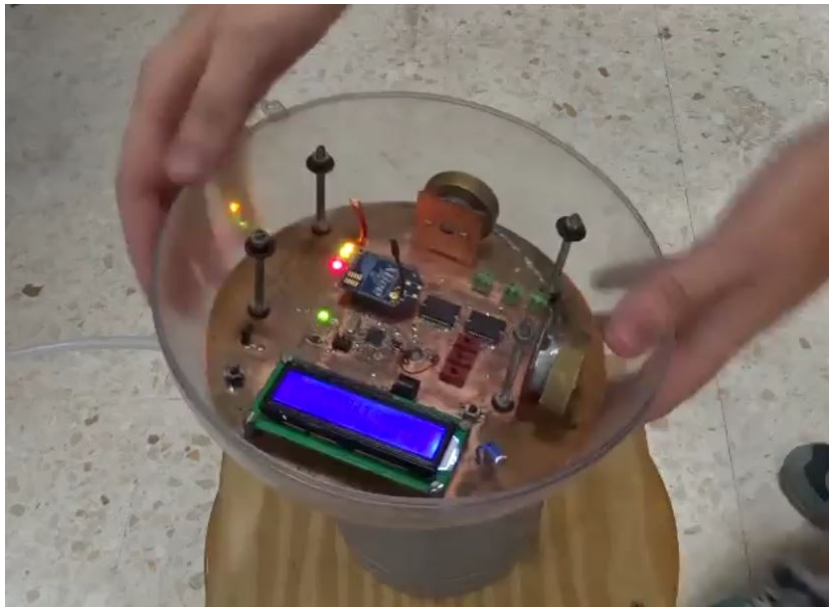
We have to say that the **PID** controller is constantly being updated (every 150 ms) but it only moves the actuators in those states in which the actuators have to turn according to the **PID** controller.

## CHAPTER

# 8

# RESULTS AND CONCLUSIONS

## 8.1 Results



**Video 8.1** – *Shake game video*

Once the [Testbed](#) has been programmed, we have to check the correct operation of both games, *shake game* and *light game*. The video [8.1](#) shows a demonstration of the *shake game*.

In this video we can see how when the **Testbed** is shaken, it automatically goes to SHAKE STATE. The sound of the buzzer indicates this transition. The **Testbed** takes as target angle the current angle when it was shaken and, consequently, it always will try to reach this angle. In the video, we can see how we put the **Testbed** in a different angle and it starts to turn until reach the target angle, doing this process twice. We can know that the **Testbed** is in the target angle because the buzzer sounds. At the end of the video, we can see in the **Testbed**'s display how when it reaches the target angle for 5 seconds, it comes back to IDLE STATE.

If there is any problem to see the video correctly, see [https://www.youtube.com/watch?v=whtYBy\\_MUro&feature=youtu.be&ab\\_channel=GranaSATTeam](https://www.youtube.com/watch?v=whtYBy_MUro&feature=youtu.be&ab_channel=GranaSATTeam).



**Video 8.2** – *Light game video*

The video 8.2 shows a demonstration of the *light game*. In this video we can see how when we illuminate one of the **LDRs**, the **Testbed** automatically goes into LIGHT STATE. The buzzer sounds and it starts to turn until the **LDR** is illuminated again. Then, the objective of the **Testbed** is to put the **LDR** facing the light source. When we remove the light source, the **Testbed** starts to turn again until it finds another light source. Although it is not shown in the video 8.2, if we wanted to come back to IDLE STATE, we would have to push the button or do the “goodbye” gesture.

If there is any problem to see the video correctly, see [https://www.youtube.com/watch?v=gEsQLZMJYRM&ab\\_channel=GranaSATTeam](https://www.youtube.com/watch?v=gEsQLZMJYRM&ab_channel=GranaSATTeam).

In the above videos we can see that the **Testbed** always starts to turn in clockwise sense. The reason is the weight distribution inside the **PCB**. If we see the **PCB** design (4.15), two of the heaviest elements (the reaction wheels) are located in the same side of the **PCB** very close to each other. In a friction-less scenario, the **Testbed** will bend down due to the gravity.

For the [Testbed](#), it is easier to turn toward the inclination is.

## 8.2 Future work

With the correct operation of the *light game* and *shake game* we can conclude the present project. In general terms, we can say that we have finished it successfully because we have achieved the client requests (2). However, there are some requirements that we have failed to achieve as well as some aspects that can be improved.

First, although we have achieved to measure the [SDR](#)' accuracy and we have obtained their accuracy curves, we could not prove and corroborate these results. We tried to get in contact with the manufacturers but we had no answer. It would be desirable to check these data in the future, in a experimentally way or achiving to get in contact with the manufacturers.

According with the [Testbed](#), it was thought to program the it using a [RTOS\(2.2\)](#). However, as we said in section 7.1.2, it has not been possible because the memory of the microprocessor was not enough. In the future, it could be a good improvement to change the current microprocessor by another one with more memory. Other possible solution could be to optimizate the code and find another [RTOS](#) wiht less memory usage.

In section 8.1 we said that the weight distribution was not uniform. This may cause problems in the [Testbed](#) performance becuase sometimes the motors do not give enough inertial moment to move the [Testbed](#) in counter-clockwise sense. It could be desirable to find a solution that equilibrate the weight.

Other problem is the use of an [air-bearing](#) to achieve the friction-less scenario. Although the [air-bearing](#) is a good election due its low cost, it only give us a friction-less scneraio in the yaw axis, and consequently, we only have one axis of freedom to orient the [Testbed](#) and it would be desirable to have a friction-less in the three axes. However, this problem has a difficult solution becuase the [GranaSAT](#) has a very limited budget.

## 8.3 Conclusions

At the begining of the present document (1.1), I spoke about my objectives and motivations to enter in the [GranaSAT](#) team and do this project. Once I have finished it, I can say that the effort has been worth it. I think that now I have more knowledge in this field what distinguishes me from other students. In the same way, all this could help me to find a job, although I am aware that I still have much to learn. Moreover, I have known the aerospace sector and I would like to have a job related to this theme.

To complete this project I had to solve many problemas I've found but, if I had to choose what was the hardest part, it was understood that the movement it is very different when we are in a friction-less scenery. Since we were in the high school, the teachers always told



us that if there is no opposing force, the movement is continuous and infinite. Although I knew that, it is now when I have really understood it, because sometimes you do not really understand something until you see it. For these reasons, when I started to program the [Testbed](#), I had numerous errors because I programmed it as if the friction-less scenario does not exist. For example, sometimes I did not take into account that when the [Testbed](#) turned in the [air-bearing](#), it rotated continuously even if we switched off the motors.

Finally, I would like to say that the best thing of this project has been the teamwork with the [GranaSAT](#) members. Moreover, I liked very much that the [Testbed](#) will be exposed in the science week of the science faculty next year. It will be an achievement if a young high school student decides to study engineering of sciences thanks to this [Testbed](#) and other similar devices.

# REFERENCES

- [1] Atmega328 pinout. Website. <http://www.r-site.net/site/struct.asp?lang=en&at=//op%5B\spacefactor\@m{id=%273080%27%5D>.
- [2] Bootloader development. Arduino Website. <https://www.arduino.cc/en/Hacking/Bootloader?from=Tutorial.Bootloader>.
- [3] Crosspack for avr development. Webpage. <https://www.obdev.at/products/crosspack/download.html>.
- [4] Drivers avr programmer & spi interface. Learn Adafruit Webpage. <https://learn.adafruit.com/usbtinyisp/drivers>.
- [5] Mpu6050 calibration. Webpage. [http://minoan.cs.nccu.edu.tw/wearable/document\\_implement/blob/233151a78425001066c009de744216e727ed8ed7/Arduino/MPU6050\\_calibration\\_v1.1/MPU6050\\_calibration/MPU6050\\_calibration.ino](http://minoan.cs.nccu.edu.tw/wearable/document_implement/blob/233151a78425001066c009de744216e727ed8ed7/Arduino/MPU6050_calibration_v1.1/MPU6050_calibration/MPU6050_calibration.ino).
- [6] Optiboot bootloader for arduino and atmel avr. Website. <https://github.com/Optiboot/optiboot>.
- [7] Overview author gravatar image - avr programmer & spi interface. Learn Adafruit Webpage. <https://learn.adafruit.com/usbtinyisp/overview>.
- [8] Reinstalando el bootloader de tu arduino. Website. <http://arduino.cl/reinstalando-el-bootloader-de-tu-arduino/>.
- [9] Watchdog timer. Webpage. [https://en.wikipedia.org/wiki/Watchdog\\_timer](https://en.wikipedia.org/wiki/Watchdog_timer).

## References

- [10] What is arduino? Arduino Website. <https://www.arduino.cc/en/guide/introduction#>.
- [11] Winavr. Webpage. <https://sourceforge.net/projects/winavr/files/>.
- [12] Features overview. Website, 2016. [http://www.freertos.org/Freertos\\_Features.html](http://www.freertos.org/Freertos_Features.html).
- [13] Freertos memory usage. Website, 2016. <http://www.freertos.org/FAQMem.html#RAMUse>.
- [14] Metodo de ziegler nichols. Website, 2016. <https://sites.google.com/site/picuino/ziegler-nichols>.
- [15] AEROSEMI. *MT3608 High Efficiency 1.2MHz 2A Step Up Converter*. <https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf>.
- [16] ALIEXPRESS. Charger moule. Website, 2016. [http://es.aliexpress.com/store/product/Free-shipping-5pcs-Lithium-Battery-Charger-Module-Board-mini-5v-USB-1A-li-1280487\\_2036149323.html?spm=2114.04010208.3.1.rdyH83&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10057\\_10056\\_10055\\_10049\\_10044\\_10043\\_10059\\_10058\\_10017\\_405\\_404\\_10041\\_10060\\_10042\\_10061\\_10062\\_412,searchweb201603\\_2&btsid=dc07f2f3-ee2c-4e77-a7ba-0238a5edcbb9](http://es.aliexpress.com/store/product/Free-shipping-5pcs-Lithium-Battery-Charger-Module-Board-mini-5v-USB-1A-li-1280487_2036149323.html?spm=2114.04010208.3.1.rdyH83&ws_ab_test=searchweb201556_0,searchweb201602_4_10057_10056_10055_10049_10044_10043_10059_10058_10017_405_404_10041_10060_10042_10061_10062_412,searchweb201603_2&btsid=dc07f2f3-ee2c-4e77-a7ba-0238a5edcbb9).
- [17] ALIEXPRESS. Dc-dc step up. Website, 2016. [http://es.aliexpress.com/store/product/1PCS-MT3608-2A-Max-DC-DC-Step-Up-Power-Module-Booster-Power-Module/2138141\\_32653449657.html?spm=2114.04010208.3.20.agDdWi&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10057\\_10056\\_10055\\_10049\\_10044\\_10043\\_10059\\_10058\\_10017\\_405\\_404\\_10041\\_10060\\_10042\\_10061\\_10062\\_412,searchweb201603\\_2&btsid=10893b4d-bb43-4d0f-8d7a-62542f37f8bf](http://es.aliexpress.com/store/product/1PCS-MT3608-2A-Max-DC-DC-Step-Up-Power-Module-Booster-Power-Module/2138141_32653449657.html?spm=2114.04010208.3.20.agDdWi&ws_ab_test=searchweb201556_0,searchweb201602_4_10057_10056_10055_10049_10044_10043_10059_10058_10017_405_404_10041_10060_10042_10061_10062_412,searchweb201603_2&btsid=10893b4d-bb43-4d0f-8d7a-62542f37f8bf).
- [18] ALIEXPRESS. Lm1117t-3.3 to-220 3.3 v 800ma regulador. Website, 2016. <http://es.aliexpress.com/item/S1041-Free-shipping-10pcs-LM1117T-3-3-T0-220-3-3V-800mA-Low-Dropout-Linear-Regulator/32353564815.html?spm=2114.43010308.4.33.FwHnXM>.
- [19] ATMEL CORPORATION. *ATmega48A/PA/88A/PA/168A/PA/328/P datasheet*. [http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-datasheet\\_Complete.pdf](http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-datasheet_Complete.pdf).
- [20] BLACK, AND DECKER. Cp2525 air compressor. Products catalogue, 2016.
- [21] BURGOS, V. Testbed for a 1u cubesat. B.s thesis (dissertation), University of Granada, Granada, September 2015. This B.S Thesis is a contribution to the GranaSAT project.
- [22] CCAM. Cubesat constellation around mars. Website, 2016. [http://ccar.colorado.edu/asen5050/projects/projects\\_2013/Naik\\_Siddhesh/Cubesats.html](http://ccar.colorado.edu/asen5050/projects/projects_2013/Naik_Siddhesh/Cubesats.html).

- [23] CONTENTCENTRAL, D. 3d contentcentral. Website, 2016. <http://www.3dcontentcentral.es/>.
- [24] CORP., N. T. P. A. *TP4056 Standalone Linear Li-Ion Battery Charger*. <https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf>.
- [25] CUBESATPRO. Cubesat standard platform. Website, 2016. <http://www.cubesatpro.com/index.php/products>.
- [26] DIGI INTERNATIONAL INC. *XBee/XBee-PRO RF Modules Datasheet*, product manual v1.xex ed. <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>.
- [27] ENERGIES, E. *Datasheet MGL2803 Sealed Lithium Ion Rechargeable Battery*. [http://www.enix-energies.co.uk/media/pdf/MGL2803\\_UK.pdf](http://www.enix-energies.co.uk/media/pdf/MGL2803_UK.pdf).
- [28] ENGINEERS, T. B. The quadcopter : control the orientation. Website, 2016. <http://theboredengineers.com/2012/05/the-quadcopter-basics/>.
- [29] EXTREME, D. Gy-511 lsm303dlhc 3-axis acceleration module. Website, 2016. <http://www.dx.com/p/gy-511-lsm303dlhc-3-axis-acceleration-gyro-module-blue-240660#.V6mhcLiLTcs>.
- [30] FUNCUBE. *FUNcube Dongle Pro+ User Manual*. <http://www.funcubedongle.com/MyImages/FCD2ManualV4.pdf>.
- [31] GITHUB. i2cdevlib. Website, 2016. <https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050>.
- [32] GRANASAT. Granasat. Website, 2016. <https://granasat.ugr.es/index.php/es/>.
- [33] GUIDE, R. Photo resistor. Website, 2016. <http://www.resistorguide.com/photoresistor/>.
- [34] I2CDEVLIB. Mpu-6050 6-axis accelerometer/gyroscope. Website, 2016. <http://www.i2cdevlib.com/devices/mpu6050#source>.
- [35] IBRAHIN, D. *Microcontroller based temperature monitoring and controll*. Elsevier Science and Technology Books, September 2002.
- [36] INSTRUMENTS, M. *Radio Communications Test Sets 2955a and 2955r*. <http://exodus.poly.edu/~kurt/manuals/manuals/Other/MARCONI%202955A,%202955R%20operating.pdf>.
- [37] INSTRUMENTS, T. *LM1117 800mA Low-Dropout Linear Regulator*. <http://www.ti.com/lit/ds/symlink/lm1117.pdf>.

## References

- [38] INTEGRATED, M. *Datasheet MAX1674/MAX1675/MAX1676 Step-up DC-DC Converters*. <https://datasheets.maximintegrated.com/en/ds/MAX1674-MAX1676.pdf>.
- [39] INTEGRATED, M. *Datasheet MAX8211/MAX8212 Microprocessor Voltage Monitors*. <http://datasheets.maximintegrated.com/en/ds/MAX8211-MAX8212.pdf>.
- [40] INVENSENSE INC. *MPU-6000 and MPU-6050 Product Specification*, revision 3.4 ed. 1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A. Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104, August 2013. [https://store.invensense.com/datasheets/invensense/MPU-6050\\_DataSheet\\_V3%204.pdf](https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf).
- [41] LARSEN, N. *BASIC WATCHDOG TIMER (Arduino UNO/ATmega328)*. [https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwj3wOag2NrOAhVEbhQKHfNHArcQFggrMAE&url=https%3A%2F%2Fforum.arduino.cc%2Findex.php%3Faction%3Ddlattach%3Btopic%3D63651.0%3Battach%3D3585&usg=AFQjCNHC1ED4HDxiLnEazSE9TXKn5JmYIw&sig2=\\_vhoU0Aya10avNC3qhhh4A&bvm=bv.129759880,d.d24](https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwj3wOag2NrOAhVEbhQKHfNHArcQFggrMAE&url=https%3A%2F%2Fforum.arduino.cc%2Findex.php%3Faction%3Ddlattach%3Btopic%3D63651.0%3Battach%3D3585&usg=AFQjCNHC1ED4HDxiLnEazSE9TXKn5JmYIw&sig2=_vhoU0Aya10avNC3qhhh4A&bvm=bv.129759880,d.d24).
- [42] LAUFER, C. *The Hobbyists Guide to RTL-SDR: Really Cheap Software Defined Radio*. <http://www.qsl.net/yo4tnv/docs/The%20Hobbyists%20Guide%20To%20RTL-SDR%20-%20Carl%20Laufer.pdf>.
- [43] LUTRON ELECTRONIC ENTERPRISE. *Photo tachometer DT-2234B*. <http://www.hestore.hu/files/DT-2234B.pdf>.
- [44] MUNOZ, J. R. P. Ardu mpu. Website, 2016. [https://www.dropbox.com/s/uqvedejtbpad3za/ARDU\\_MPU.rar?dl=0](https://www.dropbox.com/s/uqvedejtbpad3za/ARDU_MPU.rar?dl=0).
- [45] PLAYGROUND, A. Mpu-6050 accelerometer + gyro. Website, 2016. <http://playground.arduino.cc/Main/MPU-6050>.
- [46] PROCESSING. Processing. Website, 2016. <https://processing.org/>.
- [47] RTL-SDR. Rtl-sdr.com. Website, 2016. <http://www.rtl-sdr.com/>.
- [48] RTL-SDR.SCENERS.ORG. Kalibrate-rtl. Website, May 1st 2016. <http://rtl-sdr.sceners.org/?p=193>.
- [49] STMICROELECTRONICS. *LSM303DLHC:Ultra-compact high-performance eCompass module:3D accelerometer and 3D magnetometer*, November 2013. Available at <http://www.st.com/web/en/resource/technical/document/datasheet/DM00027543.pdf>.
- [50] WIKIPEDIA. Altium designer. Website, 2016. [https://en.wikipedia.org/wiki/Altium\\_Designer](https://en.wikipedia.org/wiki/Altium_Designer).
- [51] WIKIPEDIA. Buzzer. Website, 2016. <https://es.wikipedia.org/wiki/Zumbador>.

- [52] WIKIPEDIA. Cubesat. Website, 2016. <https://en.wikipedia.org/wiki/CubeSat>.
- [53] WIKIPEDIA. Pid controller. Website, 2016. [https://en.wikipedia.org/wiki/PID\\_controller](https://en.wikipedia.org/wiki/PID_controller).
- [54] WIKIPEDIA. Real time operating system. Website, 2016. [https://es.wikipedia.org/wiki/Sistema\\_operativo\\_de\\_tiempo\\_real](https://es.wikipedia.org/wiki/Sistema_operativo_de_tiempo_real).
- [55] WIKIPEDIA. Testbed. Website, 2016. <https://en.wikipedia.org/wiki/Testbed>.
- [56] WIKIPEDIA. Xbee. Website, 2016. <https://en.wikipedia.org/wiki/XBee>.

## References

## APPENDIX

# A

## BOOTLOADER INSTALLATION

In this appendix we are going to explain how to install a [bootloader](#) in a Atmega328p microcontroller. It will be followed the process described in [8]. There are two options to install a [bootloader](#). The first one is using a external programmer and the other one is using a second [Arduino](#) board for the programming. We will explain the first option. If the reader would want to intall the [bootloader](#) using the second option, see [8].

We are going to install the Optiboot [Arduino bootloader](#). It can be downloaded from [6]. Table A.1 shows the Optiboot features.

Allows larger sketches. Optiboot is a quarter of the size of the default bootloader, freeing 1.5k of extra space
Makes your sketches upload faster. Optiboot operates at higher baud rates and has streamlined programming
Adaboot performance improvements. Optiboot runs your sketches sooner, with no watchdog issues.
Compatible with 168 and 328 Arduinos including Lilpad, Pro, Nano
Believed to work with ATmega1280(“Mega”), ATmega644(“Sanguino”) and ATmega1284(“Mighty”)
Supports several additional AVR chips (ATmega88, ATmega32)

**Table A.1** – *Optiboot features* [6]



We will need an external programmer. In our case, we will use [USBtinyISP](#), whose drivers can be downloaded from [4]. Moreover, we have to install WinAVR (download it from [11]) if we are using Windows or Crosspack for AVR Development (download it from [3]) if we are using iOS.

Once we have installed the needed software, to install the [bootloader](#) we have to send the *.hex* file (included in `optiboot/optiboot/bootloaders/optiboot/` downloaded from [6]), in our case, the *optiboot\_atmega328.hex* file from the PC to the [USBtinyISP](#). For this task we will use **AVRdude**, a software included in WinAVR (Windows) and in Crosspack for AVR Development (iOS).

**AVRdude** is a software that works by commands, so we have to open a command terminal to use it. Writing “AVRdude” in the command terminal, we can see the different options to use the software. The options are described in table A.2.

Command	Description
-p <partno>	Required. Specify AVR device
-b <baudrate>	Override RS-232 baud rate
-B <bitclock>	Specify JTAG/STK500v2 bit clock period (us)
-C <config-file>	Specify location of configuration file
-c <programmer>	Specify programmer type
-D	Disable auto erase for flash memory
-i <delay>	ISP Clock Delay (in microseconds)
-P <port>	Specify connection port.
-F	Override invalid signature check
-e	Perform a chip erase
-O	Perform RC oscillator calibration (see AVR053)
-U <memtype>:r w v: <filename>[:format]	Memory operation specification. Multiple -U options are allowed, each request is performed in the order specified.
-n	Do not write anything to the device
-V	Do not verify
-u	Disable safemode, default when running from a script
-s	Silent safemode operation, will not ask you if fuses should be changed back
-t	Enter terminal mode
-E <exitspec>[,<exitspec>]	List programmer exit specifications
-x <extended_param>	Pass <extended_param> to programme
-y	Count # erase cycles in EEPROM
-Y <numeber>	Initialize erase cycle # in EEPROM
-v	Verbose output. -v -v for more
-q	Quell progress output. -q -q for less
-?	Display options

**Table A.2** – *AVRdude* command options [6]

The first step that we have to do is to configure the *Fuse* bits. These are bits that allow to configure the microcontroller at very low level. For example, we can choose if we are going to use an external clock or not or enable an external reset. To be ensure to have a correct behaviour of the microcontroller, we configure the *Fuse* bits writing the following command sentence [8]:

```
avrdude -b 19200 -c usbtiny -p m328p -v -e -U efuse:w:0x05:m -U hfuse:w:0xD6:m  
-U lfuse:w:0xFF:m
```

For more information about “Fuse” bits, see [19].

With *-b* command we indicate the baudrate (19200), *-c* indicates to use a external programmer ([USBtinyISP](#)), *-p* indicates the microprocessor that we are programming (ATmega328p), *-v* indicates to AVRdude to give a verbal output by the screen, *-e* indicates to delete the microprocessor memory before programming it and finally, *-U* indicates the writing of the “Fuse” bits (memory position and data to write). Once the “Fuse” bits have been configured, we can install the [bootloader](#) writing the following comand sentence [8]:

```
avrdude -b19200 -c usbtiny -p m328p -v -e -U flash:w:hexfilename.hex  
-U lock:w:0x0F:m
```

We have to replace *hexfilename.hex* by the name of the *.hex* file that we have to install (*optiboot\_atmega328.hex* in our case). Moreover, we write the “Lock” bits. These bits are similar to “Fuse” bits and they control the access security microcontroller. They are important because they avoid to accidentally overwrite the [bootloader](#) if we write a program to big.





## APPENDIX

# B

## KALIBRATE-RTL SOFTWARE

In this appendix it will be explained how to use [Kalibrate-RTL](#) to calculate the offset of the [RTL-SDR](#). This software can be downloaded from [48] and it is available for Linux and Windows. We will focus in the Windows version.

We must take into account that the [ppm](#) deviation of the [RTL-SDR](#) can drift with the temperature, so it is recommendable not to use the device about thirty minutes before to measure the offset. [Kalibrate-RTL](#) software uses the signal of a [GSM](#) mobile phone base station so, the first step is to scan the channels and search for the one that has the highest power. [Kalibrate-RTL](#) allows to scan in GSM850, GSM900, EGSM, DCS and PCS bands. However, DCS and PCS bands are 1800 MHZ and 1900 MHZ respectively and the antenna used (the antenna obtained with the purchase of the [RTL-SDR](#) shown in picture 3.1) is not able to work in these high frequencies. Anyway, is not necessary to scan all the bands. We only need a channel with enough power so, we only will scan the three first bands.

[Kalibrate-RTL](#) is a console application, so to use it, we have to open a Windows console and move to the directory where the software is located. With `kal [options]` command we can use all the possibilities that the software offers. The different options can be seen in table B.1.

Command	Description
-s	Band to scan (GSM850, GSM900, EGSM, DCS, PCS)
-f	Frequency of nearby <a href="#">GSM</a> base station
-c	Channel of nearby <a href="#">GSM</a> base station
-b	Band indicator (GSM850, GSM900, EGSM, DCS, PCS)
-g	Gain in dB
-d	<a href="#">RTL-SDR</a> device index
-e	Initial frequency error in <a href="#">ppm</a>
-v	Verbose
-D	Enable debug messages
-h	Help

**Table B.1** – *Kalibrate command options*

Picture [B.1](#) shows the Windows console with the [ppm](#) measurement process. It has been added some comments and it has been divided into four sections for a better understanding. Sections [1] to [3] show the scanning process of the GSM850, GSM900 and EGSM bands. It has been set a gain of 42 dB in the [RTL-SDR](#) for a better scanning and an initial offset of 22 [ppm](#)<sup>1</sup>.

In our scanning area, we only found the channels 977 and 981 of the EGSM band (section [3] of the picture [B.1](#)). As we can observe, channel 977 is a little more powerful than 981 channel, so we will use the first one for the measurement, obtaining a final result of 65.525 [ppm](#).

---

<sup>1</sup>Although we are measuring the [ppm](#) offset, we use an estimated initial offset for the scanning.

```

E:\Clases\MASTER\TRABAJO_FIN_DE_MASTER\SDRs\kalibrate-win-release>kal.exe -g 42 -e 22 -s 850
Found 1 device(s):
  0: ezcab USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcab USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 42.0 dB
meh: Scanning for GSM-850 base stations.
GSM-850:
[1] Scanning of 850 GSM band

E:\Clases\MASTER\TRABAJO_FIN_DE_MASTER\SDRs\kalibrate-win-release>kal.exe -g 42 -e 22 -s 900
Found 1 device(s):
  0: ezcab USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcab USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 42.0 dB
meh: Scanning for GSM-900 base stations.
GSM-900:
[2] Scanning of 900 GSM band

E:\Clases\MASTER\TRABAJO_FIN_DE_MASTER\SDRs\kalibrate-win-release>kal.exe -g 42 -e 22 -s EGSM
Found 1 device(s):
  0: ezcab USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcab USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
Setting gain: 42.0 dB
meh: Scanning for E-GSM-900 base stations.
E-GSM-900:
  chan: 977 (925.6MHz - 39.385kHz)      power: 55970.41
  chan: 981 (926.4MHz - 39.287kHz)      power: 54993.54
[3] Scanning of EGSM band

E:\Clases\MASTER\TRABAJO_FIN_DE_MASTER\SDRs\kalibrate-win-release>kal.exe -e 41 -c 977
Found 1 device(s):
  0: ezcab USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcab USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
meh: Calculating clock frequency offset.
Using E-GSM-900 channel 977 (925.6MHz)
average          [min, max]          (range, stddev)
- 22.701kHz      [-22761, -22651]          (110, 30.431820)
overruns: 0
not found: 2
average absolute error: 65.525 ppm
[4] Measuring offset using
      channel 977

E:\Clases\MASTER\TRABAJO_FIN_DE_MASTER\SDRs\kalibrate-win-release>

```

Figure B.1 – *PPM* measurement process with *Kalibrate-RTL*

**B**

**B**

## APPENDIX

### C

# PROJECT BUDGET

## C.1 Electronics costs

For the power supply implementation, we have needed to acquire some electronic devices and we have used the financial contribution of our main sponsor, the [ECTD](#), which paid us the electronics devices shown in the table [C.1](#).

Item	Cost(€)
Step Up	0,39€
Charger Module	0.91€
Voltage regulator	3.20€
<b>TOTAL</b>	<b>4,5€</b>

**Table C.1** – *Power Supply building cost*

Apart from the power supply, it has been necessary to add some electronic devices to the [PCB](#), which are detailed in table [C.2](#)



Item	Cost(€)
Buzzer	0,81€
Button	0,01€
Resistor 1 $\Omega$	0,02€
<b>TOTAL</b>	<b>0,84€</b>

**Table C.2** – *Other electronics devices cost*

Therefore, the total cost of the PCB implementation is broken down in the table C.3

Item	Cost(€)
Power supply	4,5€
Other electronics devices	0,84€
<b>TOTAL</b>	<b>5,34€</b>

**Table C.3** – *Total cost of the PCB implementation*

## C.2 Software

In the following table is shown how the licenses of the software were acquired:

Software	Owner of the license	Cost(€)
Arduino IDE	Antonio J. Ortiz	Free license
Altium Designer 14.3	GranaSAT	Free (sponsorship)
FreeRTOS	Antonio J. Ortiz	Free license
KeySight IO library suit	Antonio J. Ortiz	Free license
Matlab	ECTD	Free
Microsoft Visio 2013	UGR	Free (DreamSpark)
Microsoft Excel 2013	UGR	Free (DreamSpark)
Miktex	Antonio J. Ortiz	Free license
Project Libre	Antonio J. Ortiz	Free license
RS232 terminal	Antonio J. Ortiz	Free license
SDR#	Antonio J. Ortiz	Free license
SumatraPDF	Antonio J. Ortiz	Free license
TeXnicCenter	Antonio J. Ortiz	Free license
XCTU	Antonio J. Ortiz	Free license
	<b>TOTAL</b>	0

**Table C.4** – *Software cost*

### C.3 Human Resources

In this project, in every stage, a junior engineer was working during a year approximately. If we fix the salary for the junior engineer in Spain (about 10€/h), working an average of five hours a day, the total salary related to the junior engineer is:

Salary	10 €/h
Number of days	252 laboral days
Number of hours	1260 hours
<b>TOTAL</b>	<b>12600 €</b>

**Table C.5** – *Junior engineer cost*

Moreover, this project has been under the supervision of an senior engineer. If we fix the salary of a senior engineer in 50 €/h, working an average of three hours a week, the total salary related to the senior engineer is:

Salary	50 €/h
Number of weeks	52 weeks
Number of hours	156 hours
<b>TOTAL</b>	<b>7800 €</b>

**Table C.6** – *Senior engineer cost*

The total budget related to the human resources is:

Junior engineer	12600 €
Senior engineer	7800 €
<b>TOTAL</b>	<b>20400 €</b>

**Table C.7** – *Total human resources cost*