This Bachelor's Thesis addresses the mechanical,electronic design and implementation of a CanSat device. This device will be composed of two main blocks: a transmitter which is going to be launched and a receiver that will be in ground. The receiver consists of only one board whereas the transmitter is comprised with three different boards:the Main Board, the Power Board and the Uni Board.

This Bachelor’s Thesis is addressed for developing an improved CanSat aimed to become the new kit used on the CanSat Competition organized by the ESA. The improvements are meant not only for improving the kit’s efficiency but also to make a well-informed instruction manual as a way to get students to comprehend better the device’s hardware.

This wide scope requires applying professional System Engineering methodologies, which minimizes the risk and culminates with the successful

Irene Gil Martín is a Telecommunication engineer from Granada, Spain. She finished her Bachelor's Studies in 2021 at University of Granada. She started wotking in 2020 with GranaSAT Team where she worked not only on this Bachelor's Thesis but also as a member of the technical support team of the CanSat Competition 2020/2021

Andrés María Roldán Aranda is the academic head of the present project, and the student's tutor. He is professor in the Departament of Electronics and Computers Technologies

BACHELOR'S THESIS

Mechanical, Electronic Design and Implementation of a CanSaT

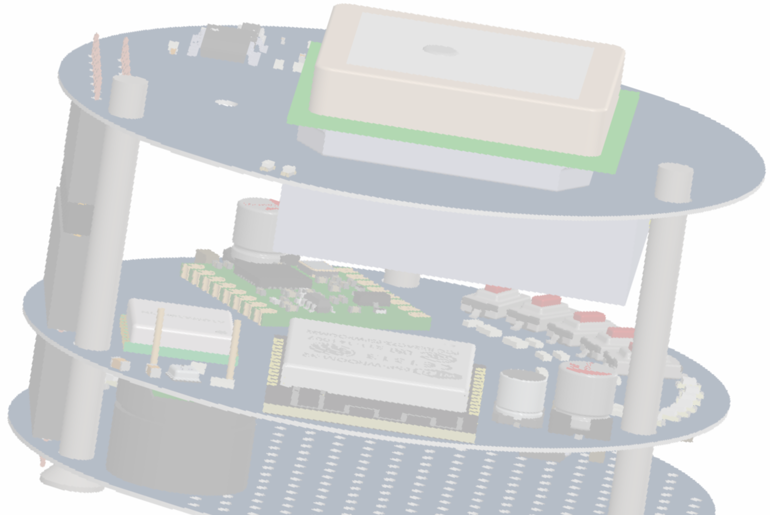
Irene Gil Martín

TELECOMMUNICATION ENGINEERING

20/21

UNIVERSITY OF GRANADA

Bachelor in Telecommunication Engineering



Mechanical, Electronic Design and Implementation of a CanSaT

Bachelor's Thesis

Irene Gil Martín

2020/2021

Tutor: Andrés María Roldán Aranda

Student Copy. Printed in Granada, September 2021.

“Mechanical,Electronic Design and Implementation of a CANSAT”

BACHELOR's DEGREE IN TELECOMMUNICATION ENGINEERING

Bachelor's Thesis

"Mechanical, Electronic Design and Implementation of a CANSAT"

ACADEMIC COURSE: 2020/2021

Irene Gil Martín

BACHELOR's DEGREE IN TELECOMMUNICATION ENGINEERING

"Mechanical, Electronic Design and Implementation of a CANSAT"

AUTHOR:

Irene Gil Martín

SUPERVISED BY:

Andrés María Roldán Aranda

DEPARTMENT:

Electronics and Computers Technologies

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Grado de D.ª Irene Gil Martín,

Informa:

Que el presente trabajo, titulado:

"Mechanical, Electronic Design and Implementation of a CANSAT"

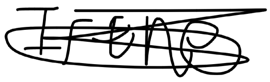
ha sido realizado y redactado por el mencionado alumno bajo mi dirección, y con esta fecha autorizo a su presentación.

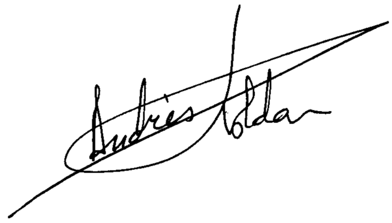
Granada, a 6 de septiembre de 2021

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 6 de septiembre de 2021





Fdo. Irene Gil Martín

Fdo. Andrés María Roldán Aranda

Mechanical, Electronic Design and Implementation of a CANSAT

Irene Gil Martín

KEYWORDS:

CanSat, CanSat Competition, ESA, Hardware, Software, Firmware Reverse Engineering, PCB, Altium Designer® 19, SolidWorks®, VS Code, PlatformIO, VNA, In-Circuit Debugger.

ABSTRACT:

This Bachelor's Thesis is addressed for developing an improved CanSat aimed to become the new kit used on the CanSat Competition organized by the ESA. The improvements are meant not only for improving the kit's efficiency but also to make a well-informed instruction manual as a way to get students to comprehend better the device's hardware. By understanding this hardware, they will be getting a closer look of the Aerospace world. It should be remarked that the Open CanSat kit, designed by Czech engineers, is a kit used in the ESA CanSat competition in all Europe. Therefore, our final and most ambitious goal is to develop a kit that would substitute the Czech kit and ideally used by all European Competitors.

The main purpose of this project is developing an improved CanSat kit able to compete in the market with a already-existing Czech CanSat kit. This kit will be composed of two differentiated blocks: a transmitter which will be introduced on the CanSat "can" and launched for measuring several parameters (temperature,pressure,etc) and a receiver which will be on the ground receiving the measures taken by the transmitter. Note that the transmitter will be composed of three differentiated PCBs while the receiver will be composed of only one PCB.

The development and implementation of this project is performed following a Reverse Engineering method giving realism and getting the student closer to professional techniques, widely recognized in the job market. Furthermore, the complexity and multidisciplinary scope of this Bachelor's Thesis allows covering not only the different specialties of the Bachelor's degree in **Telecommunication** Engineering but also acquiring knowledge and transverse abilities from other fields of the Engineering, such as **Mechanical** engineering field. Besides specific software of each of the mentioned areas, advanced techniques of **machining** (aluminum milling), **manufacturing** (solder reflow) or **characterization** of different devices (lithium batteries, antennas...) among others, have been analyzed and applied.

The result of the exposed culminates with the obtainment of a complete and

functional CanSat kit, which provides a more economic and efficient solution to the already used Open CanSat kit designed by the Czech engineers. Moreover, this Bachelor's Thesis is aimed to demonstrate the knowledge acquired by the author during the degree's academic years exhibiting the acquired knowledge about electronic and PCB design and firmware implementation (using RTOS instead of a SuperLoop architecture p.e). It should also be a proof of the student's abilities to handle tools such as Altium Designer® 19, SolidWorks®, VS Code, PlatformIO, VNA and a In-Circuit Debugger.

Even though we aim for developing a functional product, it would be a great success if the knowledge acquired during this project surpass the basic-knowledge acquired during the academic years at the expense of not having a complete functional product. Consequently, we are not focusing so much on having a ready to commercialize product (it is a long shot that would take more time than the one we have for developing this Bachelor Thesis) but in compacting all the knowledge acquired during the five years of university as well as gain the necessary skills that an engineer should have before starting to work on any company.

Electronic Design and Implementation of a CANSAT

Irene Gil Martín

PALABRAS CLAVE:

CanSat, CanSat Competition, ESA, Hardware, Software, Firmware Reverse Engineering, PCB, Altium Designer® 19, SolidWorks®, VS Code, PlatformIO, VNA, In-Circuit Debugger.

RESUMEN:

Este Trabajo de Fin de Grado tiene como objetivo el desarrollo de un CanSat mejorado destinado a convertirse en el nuevo kit utilizado en el CanSat Competition organizado por la ESA. Las mejoras están destinadas no solo a mejorar la eficiencia del kit, sino también a crear un manual de instrucciones bien informado como una forma de que los estudiantes comprendan mejor el hardware del dispositivo. Al comprender este Hardware (HW), podrán ver más de cerca el mundo aeroespacial. Cabe destacar que el kit Open CanSat, diseñado por ingenieros checos, es un kit utilizado en la competición ESA CanSat en toda Europa. Por lo tanto, nuestro objetivo final y más ambicioso es desarrollar un kit que sustituya al kit checo y que sea idealmente utilizado por todos los competidores europeos.

El objetivo principal de este proyecto es desarrollar un kit CanSat mejorado capaz de competir en el mercado con un kit CanSat checo ya existente. Este kit estará compuesto por dos bloques diferenciados: un transmisor que se introducirá en la "lata" del CanSat y se lanzará para medir varios parámetros (temperatura, presión, etc) y un receptor que estará en tierra recibiendo las medidas tomadas por el transmisor. Tengase en cuenta que el transmisor estará compuesto por tres PCB diferentes, mientras que el receptor estará compuesto por una única PCB.

El desarrollo e implementación de este proyecto se ha realizado siguiendo un proceso de Ingeniería Inversa dando realismo y acercando al alumno a técnicas profesionales, ampliamente reconocidas en el mercado laboral. Además, la complejidad y alcance multidisciplinar de esta Tesis de Grado permite abarcar no solo las diferentes especialidades del Grado en Ingeniería de Telecomunicación sino también adquirir conocimientos y habilidades transversales de otros campos de la Ingeniería, como el campo de la Ingeniería Mecánica. Además del software específico de cada una de las áreas mencionadas, se han analizado y aplicado técnicas avanzadas de mecanizado (fresado de aluminio), fabricación (reflujo de soldadura) o caracterización de diferentes dispositivos (baterías de litio, antenas ...) entre otros.

El resultado de lo expuesto culmina con la obtención de un completo y funcional kit CanSat, que aporta una solución más económica y eficiente al ya utilizado kit Open

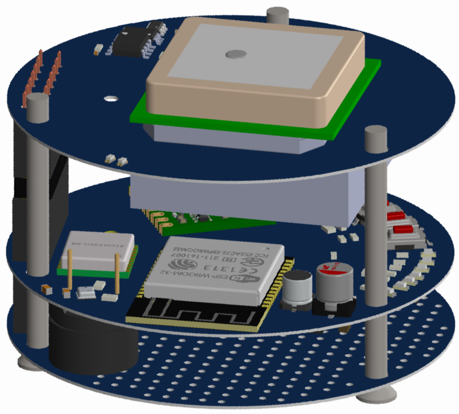
CanSat diseñado por los ingenieros checos. Además, este Trabajo de Fin de Grado tiene como objetivo demostrar los conocimientos adquiridos por la autora durante los años académicos delgrado, mostrando los conocimientos adquiridos sobre el diseño electrónico y de PCBs y la implementación de software (utilizando RTOS en lugar de una arquitectura SuperLoop p.e). También debe ser una prueba de las habilidades del estudiante para manejar herramientas como Altium Designer® 19, SolidWorks®, VS Code, PlatformIO, VNA and a In-Circuit Debugger.

Si bien nuestro objetivo es desarrollar un producto funcional, sería un gran éxito si los conocimientos adquiridos durante este proyecto superan los conocimientos básicos adquiridos durante los cursos académicos a costa de no tener un producto funcional completo. En consecuencia, no nos estamos enfocando tanto en tener un producto listo para comercializar (es una posibilidad remota que llevaría más tiempo que el que tenemos para desarrollar este Trabajo de Fin de Grado) sino en compactar todos los conocimientos adquiridos durante los cinco años de universidad, así como adquirir las habilidades necesarias que debe tener una ingeniera antes de comenzar a trabajar en cualquier empresa.

'Tough and competent'

1999 - 2021

EUROPEAN SPACE AGENCY (ESA)



Never too young for designing
your own mini "satellite"

CanSat Competition

Acknowledgments:

This work has been possible thanks to a very small amount of people, but to whom I owe a great acknowledgment; although the one I can show you in these lines will undoubtedly be insufficient, serve this brief space as such.

Firstly, I would like to thank all my family for the support they have given me since I can remember and for all the things they have done for me during all there years. For always loving me as how I am and for helping me whenever I have needed it.

Secondly, I would like to thank my friends for being always there for me and for their unconditional support. Without their encouragement I would not have made this far.

I must also thank my colleagues in the laboratory who during this year have given me their help and their technical knowledge whenever I have needed them.

Finally, I must thank my tutor, Andrés Roldán Aranda, for showing me the nice side of the Telecommunications Degree. Without his help and patient, this Bachelor's Thesis would not have been possible. It is needed many more teacher such as himself capable of bringing the students closer to the practical part of engineering and that go beyond the theoretical part.

Agradecimientos:

Este trabajo ha sido posible gracias a una cantidad muy pequeña de personas, pero a las que les debo un gran reconocimiento; aunque el que les puedo mostrar en estas líneas será sin duda insuficiente, sirva este breve espacio como tal.

En primer lugar, quiero agradecer a toda mi familia el apoyo que me han brindado desde que tengo memoria y todas las cosas que han hecho por mí durante todos estos años. Por amarme siempre como soy y por ayudarme siempre que lo he necesitado.

En segundo lugar, me gustaría agradecer a mis amigos por estar siempre ahí para mí y por su apoyo incondicional. Sin su apoyo no habría llegado tan lejos.

También debo agradecer a mis compañeros de laboratorio que durante este año me han brindado su ayuda y sus conocimientos técnicos siempre que los he necesitado.

Finalmente, debo agradecer a mi tutor, Andrés Roldán Aranda, por mostrarme el lado bonito de la Grado en Ingeniería de Telecomunicaciones. Sin su ayuda y su paciencia, este TFG no hubiera sido posible. Se necesitan muchos más profesores como él capaces de acercar a los alumnos a la parte práctica de la ingeniería y que vayan más allá de la parte teórica.

Index

Defense authorization vii

Library deposit authorization ix

Abstract xi

Dedication xvii

Acknowledgments xx

Index xxiii

List of Figures xxxiii

List of Videos xli

List of Tables xliii

Glossary xlv

Acronyms xlix

1 Introduction 1

1.1 Motivation 3

1.2 What's a CanSat? 3

1.2.1 CanSat competition 4
1.2.2 CanSat Kits 5
1.2.3 CanSat Kit Structure 6
1.3 Introduction Reverse Engineering process 7
1.4 Project Goals and Objectives 8
1.5 Project Structure 10

2 Reverse Engineering to the OpenCanSat kit 13

2.1 Product Hardware 14
2.1.1 Reviewing the Czech's Designs 14
2.1.1.1 Main Board Hardware 17
2.1.1.1.1 **Microprocessor** 19
2.1.1.1.2 **Temperature, Humidity and Presure Sensor** . . . 20
2.1.1.1.3 **MicroSD Card Socket** 22
2.1.1.1.4 **Radio Transmitter Module** 23
2.1.1.1.5 **GPS Receiver Module** 24
2.1.1.1.6 **LEDs** 25
2.1.1.1.7 **Current \Power Monitor** 26
2.1.1.1.8 **Hardware Reset** 27
2.1.1.1.9 **USB + Power Management** 28
2.1.1.2 Uni Board Hardware 30
2.1.1.3 Power Board Hardware 30
2.1.1.4 Base Board Hardware 31
2.1.1.4.1 **Microprocessor** 33
2.1.1.4.2 **MicroSD Card Socket** 33
2.1.1.4.3 **Radio Receiver Module** 34
2.1.1.4.4 **GPS Receiver Module** 34
2.1.1.4.5 **LEDs** 35

2.1.1.4.6 **TFT LCD + SD** 35

2.1.1.4.7 **Bluetooth Module** 36

2.1.1.4.8 **Hardware Reset** 36

2.1.1.4.9 **USB + Power Management** 36

2.2 Power Budget 37

2.3 Product Firmware 39

2.3.1 CanSat Transmitter firmware 39

2.3.1.0.1 Transmitter Setup Function 42

2.3.1.0.2 Transmitter Loop Function 43

2.3.1.0.2.1 INA219 tasks 43

2.3.1.0.2.2 BME280 tasks 44

2.3.1.0.2.3 GPS tasks 44

2.3.1.0.2.4 Radio tasks 45

2.3.1.0.2.5 Software RTC tasks 45

2.3.1.0.2.6 SD tasks 46

2.3.2 CanSat Receiver firmware 46

2.3.2.0.1 Receiver Setup Function 49

2.3.2.0.2 Receiver Loop Function 50

2.3.2.0.2.1 ISR Routine 50

2.3.2.0.2.2 FSM Task 50

2.4 CanSat Assembly 52

2.4.1 3D Model 52

2.5 Mass Budget 53

2.6 Validation 54

2.6.1 Transmitter Validation 54

2.6.2 Receiver Validation 57

3 System Analysis **59**

3.1 Requirements Engineering . . . 59
3.2 Electronic Architecture . . . 60
3.2.1 uProcessor Selection . . . 61
3.2.2 Temperature, Humidity and Pressure Sensor Selection . . . 63
3.2.3 MicroSD Card Socket Selection . . . 64
3.2.4 Radio Module Selection . . . 65
3.2.5 GPS Module Selection . . . 67
3.2.6 Display Module Selection . . . 69
3.2.7 Current \Power Monitor Selection . . . 70
3.2.8 LDO Regulator Selection . . . 71
3.2.9 USB Type Selection . . . 72
3.2.10 Battery Type Selection . . . 72
3.2.11 Battery Charger Selection . . . 73
3.3 Mechanical Architecture . . . 74
3.3.1 PCBs shape . . . 74
3.3.2 CanSat Can model . . . 76
3.4 Gantt Diagram . . . 77
3.5 Power Budget . . . 78

4 System Design **79**
4.1 Electronic Design . . . 79
4.1.1 Schematics Design . . . 79
4.1.1.1 Main Board Hardware . . . 80
4.1.1.1.1 **Microprocessor** . . . 81
4.1.1.1.2 **Temperature, Humidity and Pressure Sensor** . . . 86
4.1.1.1.3 **MicroSD Card Socket** . . . 86
4.1.1.1.4 **Radio Transmitter Module** . . . 87
4.1.1.1.5 **GPS Receiver Module** . . . 90

4.1.1.1.6 **LEDs** . . . 93
4.1.1.1.7 **Buzzer** . . . 96
4.1.1.1.8 **Current \Power Monitor** . . . 97
4.1.1.1.9 **Hardware Reset** . . . 99
4.1.1.1.9.1 **Reset** . . . 99
4.1.1.1.9.2 **Boot** . . . 100
4.1.1.1.10 **USB + Power Management** . . . 100
4.1.1.1.10.1 USB . . . 100
4.1.1.1.10.2 USB to Serial . . . 102
4.1.1.1.10.3 Programming Circuit . . . 104
4.1.1.1.10.4 Buck Converter . . . 107
4.1.1.1.11 Bus Connector Main Board . . . 109
4.1.1.1.12 Power Switch . . . 110
4.1.1.1.13 Engineering Mode . . . 114
4.1.1.2 Uni Board Hardware . . . 118
4.1.1.2.1 Bus Connector Uni Board . . . 119
4.1.1.2.2 LED Uni Board . . . 119
4.1.1.3 Power Board Hardware . . . 119
4.1.1.3.1 Battery . . . 120
4.1.1.3.2 Battery Charger . . . 122
4.1.1.3.2.1 TP4056 . . . 123
4.1.1.3.2.2 DW01A . . . 124
4.1.1.3.2.3 FS8205A . . . 126
4.1.1.4 Base Board Hardware . . . 126
4.1.1.4.1 **Microprocessor** . . . 127
4.1.1.4.2 **MicroSD Card Socket** . . . 127
4.1.1.4.3 **Radio Receiver Module** . . . 127
4.1.1.4.4 **GPS Receiver Module** . . . 129

4.1.1.4.5 **TFT LCD + SD** 131
4.1.1.4.6 **LEDs** 133
4.1.1.4.7 **Buzzer** 133
4.1.1.4.8 **Current \Power Monitor** 133
4.1.1.4.9 **Hardware Reset** 133
4.1.1.4.10 **USB + Power Management** 134
4.1.1.4.11 **Bus Connector Base Board** 134
4.1.1.4.12 **Power Switch** 135
4.1.1.4.13 **Engineering Mode** 135
4.1.2 PCB Design 135
4.1.2.1 Steps for a good PCB design 139
4.1.2.2 PCB module's placing 141
4.1.2.3 PCB routing 143
4.1.2.4 PCB Final Design 145
4.1.2.4.1 Transmitter Final Design 145
4.1.2.4.2 Receiver Final Design 153
4.1.3 Power Budget 156
4.1.4 Mass Budget 158
4.1.5 CanSat assembly design 160
4.1.5.0.1 3D render 164
4.2 Software Design 164
4.2.1 BME**280** Sensor Test Example 166
4.2.2 Buttons Test Example 167
4.2.3 GPS Test Example 168
4.2.4 INA**219** Test Example 169
4.2.5 LEDs Test Example 169
4.2.6 Buzzer Test Example 170
4.2.7 Radio RF**96** Test Example 171

4.2.8 SD Test Example . . . 172
4.2.9 LCD Test Example . . . 172
4.2.10 GranaSAT Complete Firmware Diagram . . . 173

5 Integration, Test and Verification 177

5.1 Final product . . . 177
5.1.1 PCB Manufactured in the GranaSAT Laboratory . . . 177
5.1.2 PCBs Industrially Manufactured . . . 179
5.1.3 PCBs soldered . . . 180
5.1.4 "Can" Assembly Production . . . 182
5.1.5 Circuits' Verifications . . . 187
5.1.5.1 USB to serial conversion . . . 187
5.1.5.2 Microprocessor and Analogue Buttons . . . 188
5.1.5.3 Power Switch . . . 189
5.1.5.4 Buzzer . . . 190
5.1.5.5 GPS Module . . . 191
5.1.5.6 LEDs . . . 192
5.1.5.7 Radio Module . . . 193
5.1.5.8 BME280 Sensor . . . 194
5.1.5.9 INA219 Module . . . 194
5.1.5.10 microSD connector and DC-DC buck converter . . . 195
5.1.5.11 DC-DC buck converter . . . 195
5.1.5.12 LCD module . . . 195

6 Conclusions and Future Lines 197

References 199

A A Project Budget 207

A.1 Materials and hardware . . . 207

A.1.1 GranaSAT CanSat Device 207
A.1.1.1 GranaSAT CanSat Can 207
A.1.1.2 CanSat Electronic Modules 208
A.1.1.2.1 GranaSAT CanSat Transmitter 208
A.1.1.3 GranaSAT CanSat Receiver 209
A.1.2 PCBs Manufacturing 210
A.2 Open CanSat vs GRANASAT cost 210
A.2.1 Open CanSat electronic cost 210
A.2.1.1 Open CanSat transmitter 210
A.2.1.2 Open CanSat receiver 211
A.2.2 Product's trade-off 212
6.3 Software 212
6.4 Human Resources Cost 213
6.5 Total Project Cost 213

B Altium Files **215**
B.1 Czech CanSat Kit SCH 216
B.2 GranaSAT CanSat Kit SCH 230
B.2.1 Main Board SCH 230
B.2.2 Power Board SCH 248
B.2.3 Uni Board SCH 250
B.2.4 Base Board SCH 251

C Simulations **271**
C.1 Diode Simulation 271
C.2 Antennas Frequency Measurement 276

D Proper Debugging **279**
D.1 Restoring Bootloader 279

D.1.1 SAMD21G18A-AUT Bootloader's Restoration 280
D.1.1.1 Materials required 280
D.1.1.2 Corrupted Bootloader Verification 281
D.1.1.3 SWD Communication Establishment 281
D.1.1.4 Bootloader Restoration 282
D.1.2 ESP32-WROOM-32D Bootloader Reestablishment 284
D.1.2.1 JTAG Communication Establishment 285
D.1.2.2 Bootloader Restoration 285

List of Figures

1.1 Product Development Process 1
1.2 Reverse engineering process 2
1.3 Porject's General Structure 2
1.4 GranaSAT Logo [8] 3
1.5 Competition logo [4] 4
1.6 Rocket Esero [4] 5
1.7 CanSat kit designed by openCanSat.eu [11] 5
1.8 Device struture 6
1.9 CanSat Transmitter [11] 6
1.10 CanSat Receiver [11] 7

2.1 Open CanSat kit 2.0 [11] 13
2.2 Open CanSat kit 2.0 Main Board [11] 15
2.3 Open CanSat kit 2.0 Power Board [11] 15
2.4 Dupont Wires [11] 16
2.5 Open CanSat kit 2.0 Uni Board [11] 16
2.6 Open CanSat kit 2.0 Base Board [11] 16
2.7 **SAMD21g18a-aut** microprocessor [29] 20
2.8 Modules Connection on the Open CanSat kit's MainBoard 20
2.9 **BMP280** sensor[22] 20
2.10 **BME280** sensor [21] 21

2.11 **BME280** sensor application circuit [21] 21
2.12 **BMP280** sensor application circuit [22] 22
2.13 502774 microSD connector[63] 23
2.14 **RFM69W** RF radio [16] 23
2.15 **MAX8C** uBLOX GPS [61] 24
2.16 **ORG12-4T-GNSS** Patch Antenna[13] 25
2.17 **INA**219 Current \Power Monitor [25] 26
2.18 **INA**219 Current \Power Monitor suggested circuit[45] 27
2.19 **Type-B micro USB** connector[48] 28
2.20 **ESD Protector** [23] 28
2.21 **LDO Regulator** [18] 29
2.22 **Li-Polymer Charge Management Controller** Module [19] 29
2.23 **Li-Polymer Charge Management Controller** Circuit [14] 30
2.24 3.7 V - 500 mA Lithium Polymeter Battery 31
2.25 Modules Connection on the Open CanSAT kit's BaseBoard 33
2.26 SMA Female Adapter Right Angle with 50 Ω Impedance 34
2.27 TFT LCD + SD Device 35
2.28 **HC-**05 **Bluetooth** to Serial Port Module [12] 36
2.29 **Open CanSat modules consumption** 37
2.30 **Open CanSat current and power needs** 38
2.31 **CH340C** chip [36] 38
2.32 **Open CanSat ON time** 38
2.33 CanSat transmitter tasks 40
2.34 PlatformIO general project structure 40
2.35 Block Diagram OpenCanSat Transmitter 41
2.36 FSM OpenCanSat Receiver 47
2.37 Block Diagram OpenCanSat Receiver 48
2.38 OpenCanSat can suggestion [3] 52

2.39 OpenCanSat modules' masses 53
2.40 OpenCanSat elements' masses 54
2.41 OpenCanSat mass 54
2.42 Open CanSat transmitter's serial monitor 55
2.43 Open CanSat transmitter's INA219.txt file 56
2.44 Open CanSat transmitter's BME280.txt file 56
2.45 Open CanSat transmitter's GPS1.txt file 57
2.46 Open CanSat transmitter's GPS2.txt file 57
2.47 Open CanSat transmitter's serial monitor 58
2.48 LCD captures after uploading the receiver software 58

3.1 Microprocessor trade-off 62
3.2 Temperature measuring modules trade-off 64
3.3 microSD sockets trade-off 65
3.4 Radio modules trade-off 66
3.5 GPS modules trade-off 68
3.6 Display modules trade-off 69
3.7 Current \Power Monitor modules trade-off 70
3.8 Buck Converter modules trade-off 71
3.9 USB connectors trade-off 72
3.10 Batteries trade-off 73
3.11 Rectangular PCB structure 74
3.12 Circular PCB structure 75

4.1 **ESP-32-WROOM-32D** microprocessor [65] 81
4.2 **ESP-32-WROOM-32D** microprocessor application circuit [65] 83
4.3 Logic Analyzer [47] 84
4.4 SPI signals (GPIO6 to GPIO12) during and after uploading a program . . . 85
4.5 **MOLEX 504528 socket [69]** 87

4.6 **RFM69 radio module [71]** . . . 88
4.7 **RFM69 Packet Structure [71]** . . . 89
4.8 **ATGM336H** module [33] . . . 91
4.9 **RFM69 Packet Structure [71]** . . . 92
4.10 **Ceramic Antenna** . . . 93
4.11 Active-Low LED circuit (I) . . . 95
4.12 Active-High LED circuit (II) . . . 95
4.13 Duble-implementation LED circuit (III) . . . 96
4.14 **KXG1205** buzzer module . . . 97
4.15 **INA219** module . . . 98
4.16 **MOLEX47346** – **0001** module [49] . . . 101
4.17 **HDMIULC6-4SC6** module [41] . . . 102
4.18 **CH340C** chip [36] . . . 102
4.19 **CH340C** implemented circuit [36] . . . 103
4.20 **2N7002 NPN MOSFET** . . . 104
4.21 DTR-*RESET*/**RTS**-*GPIO*0 during upload (tested in ESP-32S) . . . 105
4.22 DTR-*RESET*/**RTS**-*GPIO*0 simulated in Altium . . . 106
4.23 DTR-*RESET*/**RTS**-*GPIO*0 program circuit . . . 107
4.24 **AP3429/A** [31] . . . 107
4.25 **AP3429/A** implemented circuit [31] . . . 108
4.26 Bus Connector . . . 109
4.27 Power Switch Circuit . . . 111
4.28 Battery discharging in the Power Switch Circuit Altium simulation . . . 112
4.29 Button detection in the Power Switch Circuit Altium simulation . . . 112
4.30 Transistor behavior in the Power Switch Circuit Altium simulation . . . 113
4.31 Hold_Battery_On pin in the Power Switch Circuit Altium simulations . . . 113
4.32 Enable pin from the AP3429/A module in the Power Switch Circuit Altium simulation . . . 114

4.33 Circuit for detecting two buttons with only one ADC pin 115
4.34 Circuit implemented whenever the S1 button is pressed 116
4.35 Circuit implemented whenever the S2 button is pressed 117
4.36 Circuit implemented whenever the S2 button is pressed 118
4.37 **404050 Flat Li-On Battery** 120
4.38 **TP4056** charger module 122
4.39 **TP4056** suggested circuit [57] 123
4.40 **DW01A** application circuit [37] 125
4.41 **433 MHz SMA antenna** module [30] 128
4.42 SMA antenna connector 129
4.43 **FAKRA-C active antenna** module [40] 130
4.44 **ILI9341** TFT LCD + SD module [42] 131
4.45 ESP-32-WROOM-32D SCH symbol 136
4.46 ESP-32-WROOM-32D footprint 137
4.47 ESP-32-WROOM-32D 3D footprint view [38] 138
4.48 ESP-32-WROOM-32D recommended PCB land pattern [65] 139
4.49 `jlcpcb.com` manufacturer designing rules [46] 141
4.50 ESP-32-WROOM-32D recommended antenna clearance [65] 143
4.51 GPS Antenna track width 144
4.52 Radio Module Antenna track width 144
4.53 GranaSAT CanSat Main Board 3D final design 146
4.54 GranaSAT CanSat Power Board 3D final design 147
4.55 GranaSAT CanSat Uni Board 3D final design 148
4.56 GranaSAT CanSat Main Board 2D final design 149
4.57 GranaSAT CanSat Power Board 2D final design 150
4.58 GranaSAT CanSat Uni Board 2D final design 151
4.59 GranaSAT CanSat transmitter 3D final design 152
4.60 GranaSAT CanSat Base Board 3D final design 154

4.61 GranaSAT CanSat Base Board 2D final design 155
4.62 GranaSAT CanSat modules consumption 156
4.63 GranaSAT CanSat device's total consumption 157
4.64 GranaSAT CanSat buck converter power and current management 157
4.65 GranaSAT CanSat buck converter power dissipation 158
4.66 GranaSAT CanSat buck converter current consumption margin 158
4.67 GranaSAT CanSat autonomy time estimation 158
4.68 GranaSAT CanSat each module weight 159
4.69 GranaSAT CanSat each module weight 160
4.70 GranaSAT CanSat total weight 160
4.71 GranaSAT CanSat PCB and "can" weight 160
4.72 GranaSAT CanSat body "can" design (left) 161
4.73 GranaSAT CanSat body "can" design (right) 162
4.74 GranaSAT CanSat cover "can" design 163
4.75 GranaSAT CanSat base "can" design 163
4.76 Block Diagram GranaSAT CanSat Transmitter 174
4.77 State Machine GranaSAT CanSat Transmitter 175

5.1 GranaSAT CanSat Main Board PCB manufactured in the laboratory 178
5.2 GranaSAT CanSat Main Board PCB 179
5.3 GranaSAT CanSat Power Board PCB 179
5.4 GranaSAT CanSat Uni Board PCB 180
5.5 GranaSAT CanSat Main Board PCB 180
5.6 GranaSAT CanSat Main Board PCB soldered 181
5.7 GranaSAT CanSat Main Board PCB soldered 182
5.8 GranaSAT CanSat base "can" 183
5.9 GranaSAT CanSat cover "can" 184
5.10 GranaSAT CanSat body "can" (left) 185

5.11 GranaSAT CanSat cover "can" 186
5.12 Programming pin signals in the oscilloscope 188
5.13 Analogue Buttons detected 189
5.14 Power Switch oscilloscope signals 190
5.15 RS-232 signals from the GPS UART interface 191
5.16 GPS received strings 192
5.17 GPS GPGGA analyzed information 192
5.18 Radio Transmitter Test 194
5.19 Radio Transmitter Test 194
5.20 LCD Test 195

C.1 Wires Solder on a 0805 BLUE LED 272
C.2 Measuring current with an open circuit 273
C.3 Experimental BLUE Diode LED curve 273
C.4 Blue diode's ideal curve simulated on Altium 275
C.5 VNA available on the laboratory 276
C.6 Calibration connectors (from left to right: short-circuit, open-circuit and matching impedance) 277
C.7 Capture from the VNA with the matching impedance connector after calibration 277
C.8 Capture from the VNA software when measuring the GPS antenna 278
C.9 Capture from the VNA software when measuring the radio antenna 278

D.1 J-Link Device from *SEGGER* [54] 281
D.2 Corrupted Bootloader error 281
D.3 SWD J-Link interface [74] 282
D.4 CanSat transmitter connected to the SEGGER J-Link 282
D.5 Programming Device Window in ATMEL Studio 283
D.6 Programming Device Window in ATMEL Studio with microprocessor recognition 283

D.7 JTAG J-Link interface [74] 285

List of Videos

5.1 Main Board manufacturing (double click in Adobe Acrobat) 178
5.2 3D printer fabricating the ”can” (double click in Adobe Acrobat) 187
5.3 Buzzer test (double click in Adobe Acrobat) 190
5.4 LEDs test (double click in Adobe Acrobat) 193

List of Tables

1.1 Top-level objectives of the project 9

2.1 Main Board Modules Description 18
2.2 LED use and type information 25
2.3 Main Board Modules Description 33
2.4 I2C addresses of the modules using I2C communication 43
2.5 CS pin of the modules using SPI communication 43

4.1 Main Board Modules Description 82
4.2 **MicroSD Module interface to ESP-32-WROOM-32D** 87
4.3 **RFM69** main features [71] 89
4.4 **RFM69** interface [71] 90
4.5 **ATGM336H** main features [32] 91
4.6 **GPS, GLONASS** and **BDS** frequencies 92
4.7 **GPS CERAMIC ANTENNA MODULE (ACTIVE)** datasheet [13] 93
4.8 **BLUE LEDs** characteristics 94
4.9 **WHITE LEDs** characteristics 94
4.10 **RED LEDs** characteristics 94
4.11 **KXG1205 BUZZER** characteristics [15] 97
4.12 **INA219** available packages [45] 98
4.13 **MOLEX47346 – 0001** main characteristics [49] 101

4.14 **HDMIULC6-4SC6** main features [41] 102
4.15 **CH340C** main features [36] 103
4.16 **DTR**-*RESET*/**RTS**-*GPIO*0 expected signals 104
4.17 **AP3429/A** main characteristics [31] 108
4.18 **DTR**-*RESET*/**RTS**-*GPIO*0 expected signals 109
4.19 **102040 Flat Li-On Battery** main characteristics [34] 121
4.20 **TP4056** main features [57] 123
4.21 **TP4056** R_{prog} values [57] 124
4.22 **DW01A** main characteristics [37] 125
4.23 **FS8205A** main features [39] 126
4.24 **433 MHz SMA antenna** main features [30] 128
4.25 **FAKRA-C active antenna** main features [40] 130
4.26 **ILI9341 TFT LCD + SD** main features [32] 132
4.27 **ILI9341 TFT LCD + SD** Module interface to **ESP-32-WROOM-32D** 132
4.28 PCB capabilities chosen in `jlcpcb.com` [46] manufacturer 140

A.1 PLA material total cost 207
A.2 GranaSAT CanSat transmitter's cost 208
A.3 GranaSAT CanSat transmitter's cost 209
A.4 PCBs and stencil total cost 210
A.5 Open CanSat transmitter's cost 211
A.6 Open CanSat receiver's cost 212
A.7 PLA material total cost 212
6.8 Software cost 213
6.9 Human Resources Cost 213
6.10 Software cost 214

D.1 SWD vs JTAG Standard [43] 284

Glossary

EAGLE EDA software that lets PCB designers seamlessly connect schematic diagrams,component placement, PCB routing, and comprehensive library content.

0805 SMD device size 0805 mils (2013 metric): 0.08″ *x* 0.05″(2.0 mm *x* 1.25 mm).

Altium Designer® 19 EDA software used to design PCB from schematics. It allows 3D Design, as well as electronics simulation.

Arduino IDE is a Cross-Platform application that is written in functions from C and C++ used mostly for writing and uploading programs to Arduino compatible boards but also other vendor development boards.

Baud unit of measurement of symbol rate used in telecommunication and electronic which determines the speed of communication over a data channel. It is the number of distinct symbol changes (signaling events) made to the transmission medium per second in a digitally modulated signal or a baud rate line code (symbols per second or pulses per second)..

Bootloader is a program that allows you to load other programs via a more convenient interface like a standard USB cable. When you power-up or reset your microcontroller board, the bootloader checks to see if there is an upload request. If there is it will upload the new program and burn it into Flash memory. If not, it will start running the last program you loaded.

Bypass Capacitor is a capacitor that shorts AC signals to ground so that any AC noise that may be present on a DC signal is removed, producing a much cleaner and pure DC signal.

CAD Uso de computadores en el proceso de diseño y documentación de un bien o servicio.

CanSat A *can-sized* device whose mission may be to collect data,perform controlled returns, or fullfill some predetermined mission profile.

CanSat Competition Is a design-build-fly competition that provides teams (formed by high-school students) with an opportunity to experience the design life-cycle of an aerospace system.

CRC a **cyclic redundancy check is an error-detecting code commonly used in digital networks and storage devices to direct accidental changes to raw data**.

Cross-Platform is computer Software that is implemented on multiple computing platforms..

CubeSat Miniaturized satellite normally for space research, with dimensions of 1 dm^3 and mass lower than 1.33 kg per unit.

Datasheet is a document that summarizes the performance and other characteristics of a product in sufficient detail that allows a buyer to understand what the product is and a design engineer to understand the role of the component in the overall system.

Decoupling Capacitor is a capacitor used to prevent the undesirable transfer of energy from one part of an electrical network (circuit) from another.

Dupont Wires is a cable with a connector at each end, typically used to interconnect components on a breadboard, for transferring electrical signals from any part of the prototype board.

Firmware is a specific class of computer Software that provides the low-level control for a device's specific Hardware.

Footprint is the arrangement of pads (in SMT) or through-holes (in THT) used to physically attach and electrically connect a component to a PCB.

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development..

GitHub is a collaborative development platform for hosting projects using the Git version control system.

GranaSAT GranaSAT is an academic project from the University of Granada originally consisting in designing and developing 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 knowledge necessary to face an actual aerospace project.

Hardware equipment or physical support in computing that refers to the physical, tangible parts of a computer system, its electrical, electronic, electromechanical and mechanical components.

Import Wizard Altium Designer® 19 tool that will help us to convert the files from other vendors to Altium Designer files.

In-Circuit Debugger is a hardware device connected between the PC and the target microprocessor test system, and it is used to debug real-time applications faster and easier.

J-Link is the name for a SEGGER device that supports directly interfacing SPI Flashes (directly communicating via the SPI protocol). It comes with a set of speed-optimized,built-in flashloaders.

Knots is a unit of speed equal to one nautical mile per hour,exactly 1.852 km/h.

MCU is a small computer on a single metal-oxide-semiconductor (MOS) IC.

Microprocessor any type of miniature electronic device that contains the arithmetic,logic and control circuitry necessary to perform the functions of a computer's CPU. This kind of IC can interpret and execute program instructions as well as handle arithmetic operations.[2].

Mil A thousand of an inch is a derived unit of length in a system of units using inches. Is a measurement unit most commonly used in engineering and manufacturing in non-metric countries.(1 mil = 0.0254 mm).

MQTT is a lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, loss-less, bi-directional connections can support MQTT.

Multimeter is a measuring instrument that can measure multiple electrical properties such as the voltage, resistance, current of farads.

PlatformIO PlatformIO is a cross-platform, cross-architecture, multiple framework, professional tool for embedded systems engineers and for software developers who write applications for embedded products. It implements a rapid and embedded development (C/C++) platform without external dependencies and a unified debugger..

Reverse Engineering Is a process or method through the application of which one attempts to understand through deductive reasoning how a device, process, system, or piece of software accomplishes a task with very little (if any) insight into exactly how it does so.

Schottky Diode is a semiconductor diode formed by the junction of a semiconductor with a metal. It has a low forward voltage drop and very fast switching action.

Shunt Resistor is a device that creates a low-resistance path for electric current, to allow it to pass around another point in the circuit.

SoC Circuito que integra todos los componentes dentro de un chip. Todas las partes analógicas, digitales, mixtas e incluso las de radio frecuencia se fabrican en el sustrato de una oblea semiconductora con la posiblidad de usar tecnologias de fabricación diferentes en cada bloque.

Software is a collection of instructions and data that tells a computer how to work.

SolidWorks® Software from Dessault Systèmes for 3D Mechanical Design.

Spreadsheet is a computer application for organization, analysis, and storage of data in tabular form.The most known is Microsoft Excel.

Stencil is a sheet of stainless steel with laser-cut openings used to place some solder paste on a PCB board for surface mount component placement.

SWD is the protocol designed by ARM for programming and debugging their microcontrollers. It also comes with many special features like sending debug info to the computer via the IO line.

Task set of program instructions loaded in the memory.

Threads unit of CPU utilization with it own program counter and stack.

Acronyms

1PPS 1 Pulse Per Second.

AC Altern Current.

ADC Analog to Digital Converter.

AFH Adaptive Frequency Hoping Feature.

ARM Advanced RISC Machine.

ASCII American Standard Code for Information Interchange.

BDS BeiDou Navigation Satellite System.

BJT Bipolar Junction Transistor.

CDR Critical Design Review.

cm Centimeters.

CMOS Complementary metal-oxide-semiconductor.

CPU Central Processing Unit.

CS Chip Select.

CSR Certificate Signing Request.

DAC Digital to Analog Converter.

DC Direct Current.

DGPS Differential Global Positioning System.

DOP Dilution of Precision.

DR Dead Reckoning.

DTR Data Terminal Ready.

EAGLE Easily Applicable Graphical Layout Editor.

EDA Electronic Desing Automation.

EEPROM Electrically Erasable Programmable Read-Only Memory.

ESA European Space Agency.

ESD Electrostatic Discharge.

Esero European Office of Resources for Soace Education in Spain.

ESR Equivalent Series Resistance.

FET Field Effect Transistor.

FIFO First In First Out.

FSK Frequency Shift Keying Modulation.

FSM Finite-State Machine.

FTP File Transfer Protocol.

GFSK Gaussian Frequency Shift Keying Modulation.

GGA Generalized Gradient Approximation.

GLONASS Global Navigation Satellite System (Russian).

GMSK Gaussian Minimum Shift Keying Modulation.

GNSS Global Navigation Satellite System.

GPALM GPS almanac data.

GPGGA GPS fix data.

GPGLL GPS antenna position data.

GPGRS GPS range residuals.

GPGSA GPS DOP (dilution of precision) and active satellites.

GPGST GPS pseudorange statistics.

GPGSV GPS satellites in view.

GPI General Purpose Input.

GPIO General Purpose Input Output.

GPMSS Beacon receiver signal status.

GPRMC Recommended minimum specific GPS data.

GPS Global Positioning System.

GPVTG Course over ground and ground speed.

GPZDA GPS time and date.

HDOP Horizontal Dilution of Precision.

HSPI High-Speed Serial Peripheral Interface.

HTTP Hypertext Transfer Protocol.

HW Hardware.

I2C Inter-Integrated Circuit.

I2S Integrated Interchip Sound.

IC Integrated Circuit.

IP Internet Protocol.

ISM Industrial, Scientific and Medical (Radio-bands).

ISR Interrupt Service Routine.

JTAG Join Test Action Group.

LCD Liquid-Crystal Display.

LDO Low Drop-Out.

LED Light-Emitting Diode.

LIN Local Interconnect Network.

LNA Low-Noise Amplifier.

m Meters.

MISO Master In Slave Out.

mm Millimeters.

MOS Metal-Oxide Semiconductor.

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor.

MOSI Master Out Slave In.

MSK Minimum Shift Keying Modulation.

MSL Mean Sea Level.

NMEA National Marine Electronic Association.

OOK On-Off Keying Modulation.

OS Operating Sytem.

OTA Over-the-Air.

PC Personal Computer.

PCB Printed Circuit Board.

PCM Protection Circuit Module.

PDR Preliminary Design Review.

PGA Programmable Gain Amplifier.

PLA Polymerized Lactic Acid.

PTC Peripheral Touch Controller.

PTH Plated Through Holes.

PWM Pulse Width Modulation.

QZSS Japan Quasi-Zenith Satellite System(Japanese).

RAM Random Access Memory.

RF Radio Frequency.

RISC Reduced Instruction Set Computer.

ROM Read Only Memory.

RSSI Received Signal Strength Indicator.

RTC Real Time Clock.

RTOS Real-Time Operative System.

RTS Request to Send.

SBAS Satellite Based Augmentation System.

SCH Schematic.

SCL Serial Clock Line.

SCLK Clock.

SD Secure Digital.

SDA Serial Data Line.

SDIO Secure Digital Input Output.

SF Spreading Factor.

SMA SubMiniature version A.

SMBUS System Management Bus.

SMD Surface Mount Devices.

SMT Surface Mount Technology.

SPI Serial Peripheral Interface.

SPL Sound Pressure Level.

SRAM Static Random Access Memory.

SS Slave Select.

SSL/TLS Secure Socket Layer / Transport layer Security.

TCP Transmission Control Protocol.

TFT Thin-Film-Transistor.

THT Through-Hole Technology.

UART Universal Asynchronous Receiver-Transmitter.

UDP User Datagram Protocol.

USART Universal Synchronous/Asynchronous Receiver Transmitter.

USB Universal Serial Bus.

UTC Coordinated Universal Time.

VAT Value Added Tax.

VNA Vector Network Analyzer.

VS Code Visual Studio Code.

VSPI Very High-Speed Serial Peripheral Interface.

WiFi Wireless Fidelity.

Chapter 1

Introduction

This Bachelor's Thesis is presented as a compilation of the knowledge acquired throughout the years of the bachelor's degree and specially, during this project period. It aims to reflect the engineering process behind the design, development, prototyping and verifying stage of a product. The overall goal of the project is developing an improved alternative of the already existing CanSat . This project has been performed seeking to better understand the process used in the engineering industry to launch a product that competes with another that already exists. In this process, we do not only get closer to the supply and demand concept in the industry but we also comprehend better the steps needed for developing any product Figure 1.1.

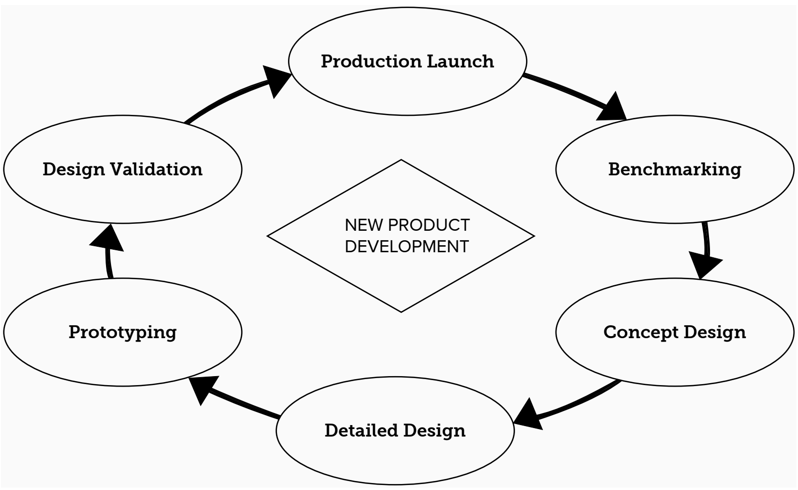


Figure 1.1 – *Product Development Process*

In order to develop the project, we have performed a reverse engineering process to the existing device, focusing not only on the hardware of the device but also on the device's software. This process (Figure 1.2) requires performing the whole process in-between the consumer and the developer (in this particular order). The final goal of this process is to create a similar product improving, not only it's qualities but also the price, to be able to compete on the market against the existing one.

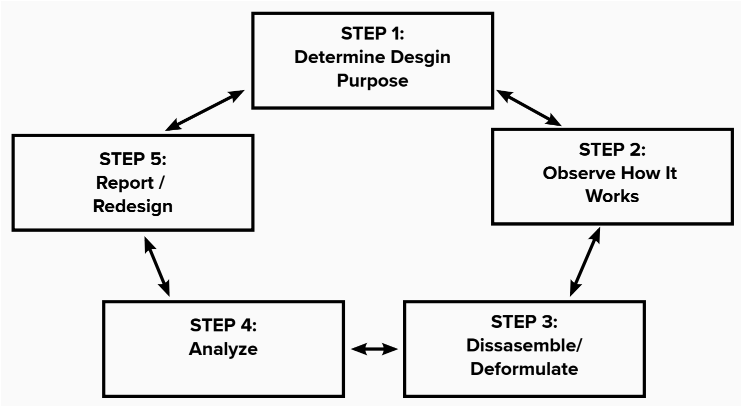


Figure 1.2 – *Reverse engineering process*

For keeping a well-organized structure, the project has been developed dividing the reverse engineering process as showed on Figure 1.3

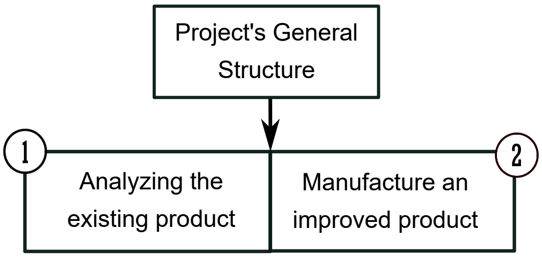


Figure 1.3 – *Porject's General Structure*

This document also follows that division, which eases understanding and allows a natural progress to the reader.

This Bachelor's Thesis fits within GranaSAT Project [8], a multidisciplinary project which gathers people from a variety of fields who are committed to acquiring new knowledge related to Electronics and Aerospace Engineering. Since its origins, one of its main purposes has been getting a CubeSat [1] in orbit ; however, today its goals goes far beyond, and a wide range of devices and projects are developed in collaboration with different students and enterprises.

Figure 1.4 – *GranaSAT Logo [8]*

1.1 Motivation

The main motivation of this project is to present a better CanSat kit by improving the device's efficiency as well as improving the product price. Developing a well-informed instruction manual is another goal of this project so that students are able to better understand the product's *hardware*.

By achieving the project objective, it will be met my own motivation. The motivation that let me develop this project was to acquire knowledge about how the industry actually works. Beside that, I aimed to get the confidence of being able to apply all the theoretical knowledge acquired through my five years of college.

1.2 What's a CanSat?

CanSat is an initiative of the ESA [6] that challenges students from all over Europe to build and launch a mini "satellite" the size of a soda can. The challenge for students is to adapt all the major subsystems found in a satellite, such as a device power system, magnitude measuring sensors, and a communication system, into the volume and shape of a soda can. The CanSat is then launched by a rocket to an altitude of approximately one kilometer, or launched from a platform, drone or captive balloon. Then their mission begins. It involves conducting a science experiment and / or technology demonstration, achieving a safe landing, and analyzing the collected data.

The CanSat competition offers a unique opportunity for students to get a closer look to a real space project. This project will not only allow them to improve their teamwork skills, sense of responsibility and creativity but also will allow them to get a closer look to the aerospace,electronics and computer engineering.

1.2.1 CanSat competition

Spain organizes a national competition, through Esero [4], to choose the team that will participate at the European championship. Participation is open to teams of 4-6 students from 14 to 19 years old, coordinated by a teacher or mentor. Participants must be from a ESA Member State, Canada, Latvia, Slovenia or Malta and be enrolled in a secondary school. The technical requirements to build the CanSat are those established by ESA in the bases of that year's competition [5].

Figure 1.5 – *Competition logo [4]*

All teams must build their own CanSat or use one CanSat kit and program it to fulfill two missions:

* **Primary Mission:** The team must measured the air temperature and the atmospheric pressure after launching the CanSat and during its descent. This parameters measurement must be transmitted by telemetry to the earth station at least once per second as well as saved on a microSD in case the communication failed. During the post-flight study, the team should be able to analyze the data obtained (for example, calculate the altitude) and capture them in graphs.

* **Secondary Mission:** This is up to the team's choices. It may be based on ideas drawn from real satellite missions, on the capture of scientific data for a specific project, on a technological demonstration of a component designed by the students, or on any other mission that fits inside the CanSat and shows its capabilities. Above all this, this mission must demonstrate that it has some scientific, technological or innovative value.

Beside meeting this two missions, they must must deliver a CDR report and a PDR within the established deadlines so that the judges can score them.

After delivering the PDR, the teams are gathered on a remote location for the competition's final step: **launching the CanSat**. As we have mentioned, the CanSat is designed to be launched from a 1000 m altitude. For reaching this alitude, Esero has designed a rocket illustrated in Figure 1.9 [7].

Figure 1.6 – *Rocket Esero [4]*

The winner team is selected not only by selecting the most ambitious missions, but for their correct development as well as a good PDR report with a proper promotion of their project on their town.

1.2.2 CanSat Kits

The CanSat kits are build to fulfill the **Primary Mission** and has all the resources needed to include all modules needed to achieve the **Secondary Mission**. They are designed for those teams with no resources or with little technical knowledge.

The CanSat kits can be build with only the modules needed to achieve the **Primary Mission** (radio module and a temperature and pressure module) or with some extra modules to fulfill the **Secondary Mission** (for example a GPS, humidity sensor, etc).

As for the CanSat kit we will be studying, it has some extra modules such as gps or humidity sensor which will led students to do a little secondary mission without needing to add any other modules. In case the team's students are confident, they are able to add new modules thanks to the pin headers installed for achieving a more ambitious mission.

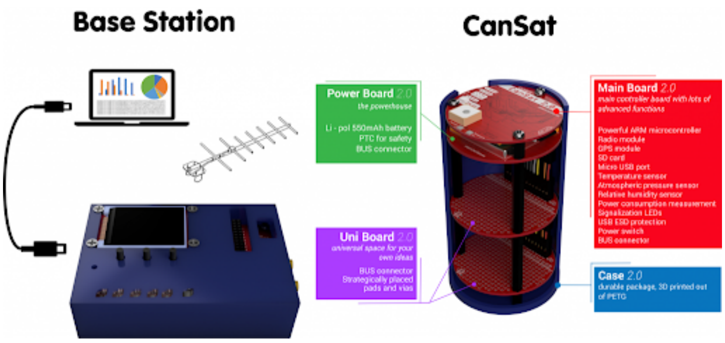


Figure 1.7 – *CanSat kit designed by openCanSat.eu [11]*

1.2.3 CanSat Kit Structure

1

Whether the project device has been developed from scratch by the team students or already manufactured, all project's devices must consists of two different structures: the **transmitter** and the **receiver** as illustrated on Figure 1.8.

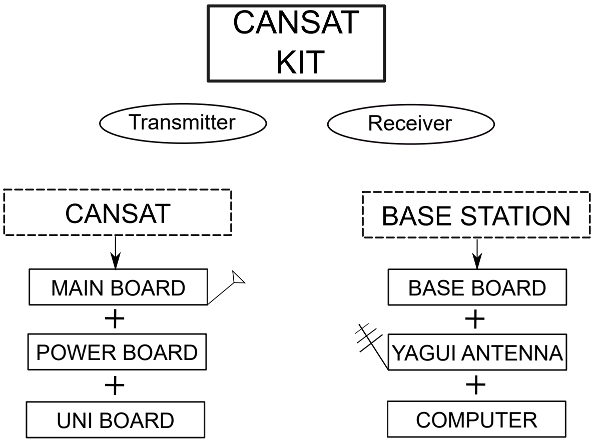


Figure 1.8 – *Device struture*

- **CanSat (transmitter):** the main subsystem of a CanSat project. Is the one introduced on a can and launched to a 1000 m altitude for developing the two missions (measuring several parameters and transmitting them to the ***base station***). It is composed of, at least, 3 different boards:
  - Main Board: Printed Circuit Board (PCB) containing, at least, the microprocessor. All extra modules fitting into the PCB shall be added.
  - Power Board: PCB containing the CanSat power system.
  - Uni Board: PCB for introducing the extra modules that did not fit into the *Main Board*. It is possible to add as many PCB for extra modules as the can could lodge.

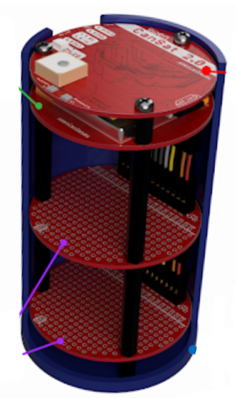


Figure 1.9 – *CanSat Transmitter [11]*

- **Base Station (receiver):** It is in charge of establishing communication with the CanSat for receiving all the measurements. It is needed for monitoring the mission's successfulness. It is composed of three modules:
  - Base Board: Printed Circuit Board (PCB) (Figure 1.10) containing, at least, the microprocessor and the radio module. It is the one echarged of communicating with the *CanSat* and receiving all the information from it. It can also lodge some other modules for measuring some other parameters such as the location in which the competition is taken place or any other we design to measure.
  - Yagui Antenna: Antenna needed for succesfully establishing the communication between the *transmitter* and the *receiver*.
  - Computer: In charge of displaying all the measurements received from the CanSat on the monitor. The connection between the *Base Board* and the computer is made through the Serial Port.

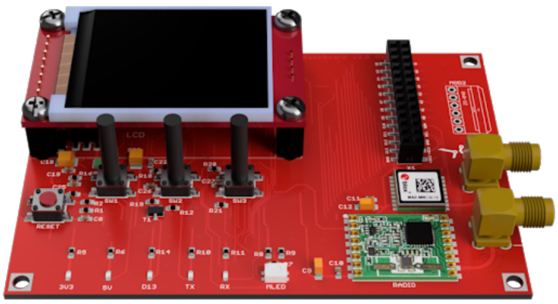


Figure 1.10 – *CanSat Receiver [11]*

1.3 Introduction Reverse Engineering process

Reverse engineering [10] is the process of duplicating an existing component, subassembly, or product without having the original drawings, documentation, or computer models. This process aimed to deduce how a product works by moving from the user's point of view to the designer's point of view.

Reverse engineering is very common in such diverse fields as software engineering, entertainment, automotive, consumer products, microchips, chemicals, electronics, and mechanical designs.

This process can be systematized by following the subsequent steps:

- Observe and assess the mechanisms that make the device work.
- Dissect and study the inner workings of a mechanical device.
- Develop the part or product geometry in a model from the actual part or product.

- Compare the actual device to your observations and suggest improvements.

1

1.4 Project Goals and Objectives

This section outlines the main top-level non-technical goals of the project. The reason to limit this section to those is to stick to the **Reverse Engineering Process** described in Section 1.3 and followed throughout this bachelor's Thesis.

- **Information Collection:** All related information about the product (hardware and software) was collected in this steps. We found not only the Altium project but also some software test of the modules [3].
- **Hardware Information Study:** In order to get familiarity with the system the information collected in the first step was studied. By doing this, we created a new Altium project, including extra information about the modules. During this step, we gained the theoretical knowledge of the hardware part of product.
- **Hardware Product Study:** After analyzing the hardware part of the product from a theoretical point of view, we analyzed the final product by observing the product. During this step, we gained the practical study of the product.
- **Software Information Study:** Once we have studied the hardware part of the product, we needed to study the software part of the product. By doing this, we first studied the test we found on internet for the diverse modules understanding how the modules worked on separately.
- **Firmware Product Study:** During this step we created a firmware for the existing product for performing the missions for which it was created.
- **Hardware Product Improvement:** Once we studied the existing product, we were prepared for presenting an improved product, with more suitable modules. When making a change, we were thinking not only on the efficiency improvement, but also in improving the price (cheaper modules).
- **Firmware Product Improvement:** In this step, we migrated the firmware developed on the *Firmware Product Study* step to be suitable for the new modules.

Ref.	Objective
Obj.1	To learn what's a CanSat, what are the requirements of the CanSat competition as well as the bases for this competition.
Obj.2	To collect all the information related to the device's hardware and firmware for the Czech CanSat Kit.
Obj.3	To understand the hardware of the Czech CanSat kit
Obj.4	To develop an Altium project of the Czech Kit with all the module's relevant information.
Obj.5	To develop the firmware for the Czech Kit to see how it works.
Obj.6	To presenting an improved Altium project with my suggestions based on improving the efficiency and price of the kit.
Obj.7	To design a prototype to test the improvements and if the modules are well chosen.
Obj.8	To migrate the firmware designed for the Czech kit to the new improved kit.
Obj.9	To design a final kit with all the improvements.
Obj.10	To demonstrate the knowledge acquired during the Bachelor's degree in Telecommunication Engineering, as well as multidisciplinary abilities gathered during the execution of this Bachelor's Thesis.
Obj.11	To successfully overcome the subject 'Bachelor's Thesis'

Table 1.1 – *Top-level objectives of the project*

Therefore, objectives listed in Table 1.1 must be understood as the author's expected results in academic and professional terms of the execution of this project.

1

1.5 Project Structure

This project, divided into six chapters and an addendum, progressively analyzes the system under development from different points of view, addresses the design tasks and finalizes with the successful completion of the product.

These chapters are:

- **Chapter 1 : Introduction**. This chapter, which is intended to be an introduction, show the general objectives and the reasons which motivate this project. The prior art, as well as a brief introduction to the system engineering methodology applied in this development, are also included in this chapter.
- **Chapter 2: Reverse Engineering to the OpenCanSat Kit**. The second chapter addresses the five first steps presented on the Figure 1.2. This chapter will lead us to the perfect understanding on their kit as it will be presented the full product from the SCH to the device's tests. In addition, it is intended to find the weaknesses of the product to be able to improve them in the Chapter 3.
- **Chapter 3: System Analysis**. According to the system engineering methodology applied, the third stage of the project corresponds to the **System Analysis**, so it is treated in this chapter. It is included a short requirements redefinition to not lose the track of this Thesis. On this chapter we will be trading-off the different modules possibilities and selecting the right fit for our device. On the last section of this chapter we will be choosing the perfect PCB shape as well as introducing the "can" design.
- **Chapter 4: System Design**. The fourth chapter deals with **system design**. It translates the technological solutions analyzed in the previous chapter to actual systems able to execute the tasks required. The blocks structure of the project introduced in the previous chapter is followed again, and each of them is extensively treated, including details at all levels of the design task.
- **Chapter 5: Integration, Test and Verification**. Once the design is completed, chapter five addresses a series of validation tests, in order to check that the system meets the **Formal Requirements** defined and, consequently, the Functional Requirements. It is worth noting that this validation is performed using one of the tools developed within this project.
- **Chapter 6: Conclusion and Future Lines**. Finally, chapter six includes the main conclusions extracted from this Bachelor's Thesis, as well as some future lines of work which have naturally emerged during the design process.
- **Appendix A: Project's Budget**. In this first Appendix will be included a detailed budget and associated cost of this project.
- **Appendix B: Altium Files**. The SCH files from the Altium projects we have developed during this Bachelor's Thesis are included in this appendix.
- **Appendix C: Simulations**. This appendix includes the procedure for simulating a

LED and extracting its simulation model for Pspice. It is also included the steps needed for measuring RF parameters with a VNA.

- **Appendix D: Data Analysis**. This last appendix comprehends the sections for analyzing a GPS string and also the steps needed for restoring the bootloader.

Chapter 2

Reverse Engineering to the OpenCanSat kit

As we have mentioned on Chapter 1, during this chapter we will be performing a reverse engineering process to the **open CanSat kit 2.0** [11] (Figure 2.1) designed by a group of four *Czech Engineers*. We are already in possession of the product's main objective, complete the two mission described on Subsection 1.2.1, but we do not know how they are going to be achieved. This means that we have an user's level information but not the technical knowledge. This chapter is meant to give us the product's technical knowledge.

Figure 2.1 – *Open CanSat kit 2.0 [11]*

For that reason, we will be dividing this chapter into two main parts: the **product hardware** and the **product software**. As well as this two main sections, we will be including some sections to include the CanSat assembly, the power and mass budget and the validations.

Before embarking on the analysis, we need to remember that during this chapter we intend to address the first four steps of the reverse engineering process described in the Section 1.4.

2

2.1 Product Hardware

By analyzing the product hardware, we are not only going to learn the product's technical specifications but we are also going to analyze the possibles weakness of the product. Achieving both purposes, we are going to be able to develop a new proposal of an improved product.

Therefore, first we will download all the available hardware designs from the ***Czech Engenieers's GitHub*** [3] studying their modules and its connections. After reviewing their information, we will be developing a new Altium project. On this new project we will add the information we believe is most relevant to the modules, as well as introduce improvements for a design's better understanding.

2.1.1 Reviewing the Czech's Designs

Their designs are on Eagle which is a software similar to Altium. As for them, we do not know why they have chosen Eagle over Altium, but for us, the reason is as simple as that Altium is much more easier to use for beginners. [9]

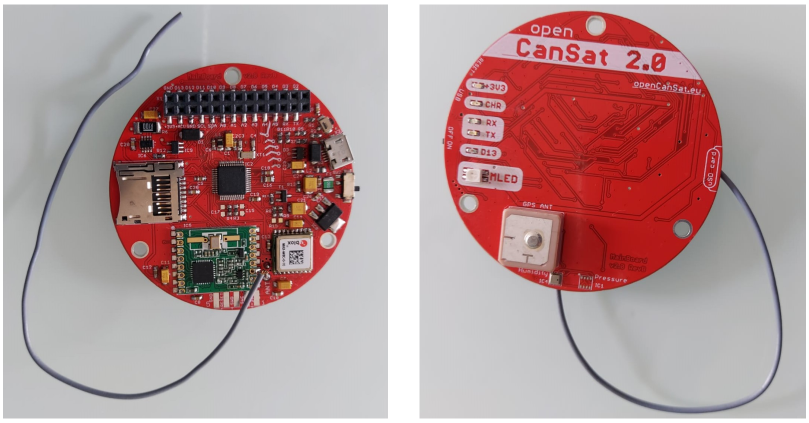
Thanks to the Altium's Import Wizard we are able to convert the Eagle sources into Altium files.

By analyzing their files, we have seen that their have divided their project into four different boards:

- **Main Board:** this board (Figure 2.2) contains the CanSat transmitter. This board is formed by the microprocessor and all the transmitter's functioning modules to succesfully fulfill the *Primary Mission* and a simple *Secondary Mission*.
- **Power Board:** this board (Figure 2.3) contains the transmitter's battery circuit as well as a header pin to connect it to the ***Main Board*** with the *dupont wires* (Figure 2.4). Without this board, the ***Main Board*** will not be able to work.
- **Uni Board:** this board (Figure 2.5) not only contains a LED but also a great amount of PTH. Therefore, it is possible to add new modules to fulfill a much

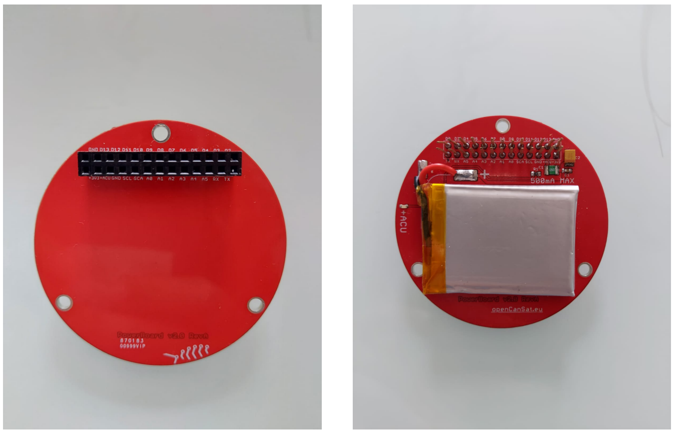
more ambitious *Secondary Mission*.

- **Base Board:** this board (Figure 2.6) contains the CanSat receiver. This board is formed by the Microprocessor and all the receiver's functioning modules to succesfully fulfill the *Primary Mission* and a simple *Secondary Mission*.



(a) *Main Board's top view* **(b)** *Main Board's bottom view*

Figure 2.2 – *Open CanSat kit 2.0 Main Board [11]*



(a) *Power Board's top view* **(b)** *Power Board's bottom view*

Figure 2.3 – *Open CanSat kit 2.0 Power Board [11]*

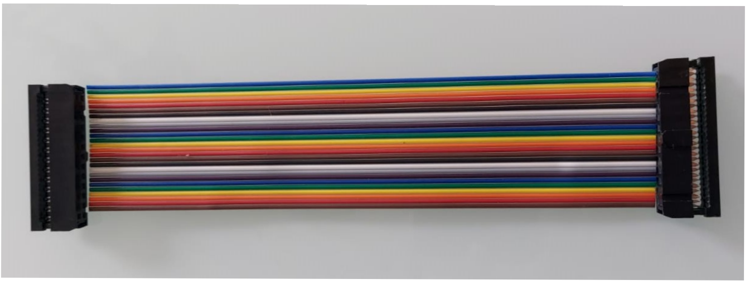
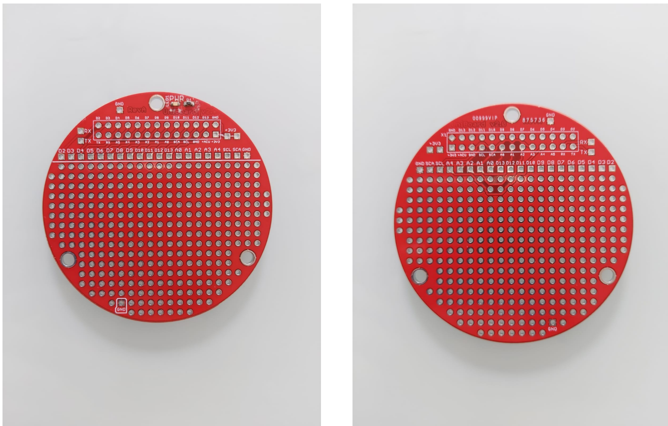
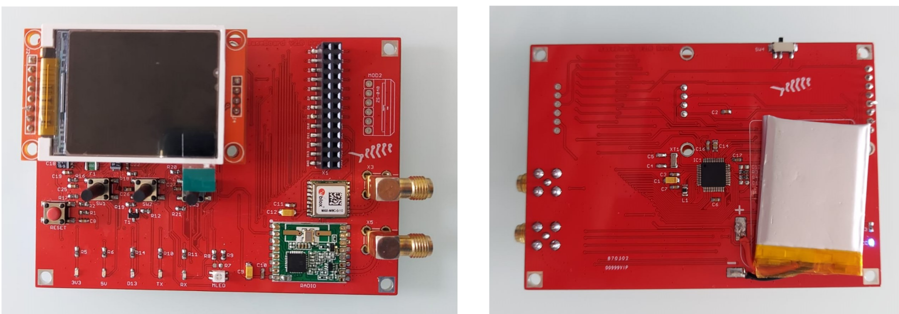


Figure 2.4 – *Dupont Wires [11]*



(a) *Uni Board's top view* **(b)** *Uni Board's bottom view*

Figure 2.5 – *Open CanSat kit 2.0 Uni Board [11]*



(a) *Base Board's top view* **(b)** *Base Board's bottom view*

Figure 2.6 – *Open CanSat kit 2.0 Base Board [11]*

Being introduced the different boards that the open CanSat kit 2.0 contains, we are going to list all the modules that make up the different boards by analyzing their project's schematics (in the future text we will refer to them as SCH) and observing the real product.

2.1.1.1 Main Board Hardware

By simply observing the board (Figure 2.2) we can identify the microprocessor on the middle, an USB connector, a microSD socket, a pin header, a switch and some resistors, capacitors and LEDs.

When studying the Eagle's SCH project, we can gather the following information about the modules extracted from their datasheet:

Module	Model	Information
Microprocessor	SAMD21G18A-AUT	Device's brain. Microprocessor from *SAMD 2X Series Microcontrollers Family* in charged of executing all the module's programs
GPS	uBLOX-MAX8C	The GPS is in charged of locating the device in case the landing goes wrong. It can also measure some interesting parameters as the device velocity
Ceramic Patch Antenna	ORG12-4T-GNSS	A 1575 ± 3 Hz axial ratio antenna needed for synchronizing the GPS module
Radio	RFM69W	Transmitter module needed for sending the data to the ***Base Station*** with a 433 MHz frequency
Antenna Cable	Electronic Wire	A 17 cm electronic wire performing as the antenna's Radio
Pressure and Temperature Sensor	BMP280	A sensor for measuring the pressure and temperature parameters needed to fulfill the ***Primary Mission***
Humidity and Temperature Sensor	BME280	A sensor for measuring the humidity and temperature parameters useful for fulfilling a simple ***Secondary Mission***
MicroSD socket	MX-502774-0891	A microSD socket for connecting an SD to save the measurements. This module is needed for obtaining the measurements in case the communication between the *transmitter* and *receiver* is down

Continued on next page

2

Module	Model	Information
DC Current Measurement	INA219	A module for measuring the device's consumption. It can measure the current and the voltage through the Shunt Resistor
Programming Pins	SAMTEC-A-TSM	A 5x1 Join Test Action Group (JTAG) port with a clock + single bi-directional data pin, providing all the normal JTAG debug and test functionality plus real-time access to system memory without halting the processor or requiring any target resident code)
USB Connector	type-B microUSB	A microUSB connector for establishing the communication with the computer through the serial port and for charging the module
Electrostatic Discharge ESD Protector	USBLC6-2SC6Y	An ESD protector to prevent the possible surges derived from the human's board manipulation
Battery Charger	MCP73831	A module for charging the device through the USB. From this module we will also take the $+5\ V$ voltage supply
LDO Regulator	MCP1826S	A module used for converting the battery's 3.7 V tension into 3.3 V tension used to power the different modules
Main Bus Connector	13x2 female pin header	A pin header to connect this board to the **Power Board** and to the **Uni Board**

Table 2.1 – *Main Board Modules Description*

Note that all the modules they have used are powered with 3.3 V tension. The only device that is powered with 5 V instead of 3.3 is a LED. Consequently, they have introduces a boost converter only for a LED which does not make much sense.

After understanding their altium project, we developed our own Altium project where we included not only the connection but also the most relevant information about the modules. This project can be found on Section B.1.

When creating the new Altium project, we have better known the need to use the hardware that they have implemented. We are going to explain briefly the knowledge acquired during this period.

It should be mentioned that, even though we have included on the diagram and on the SCH the $BMP280$ module,the Bluetooth module and the programming pin circuit, we have to mention that there is a footprint on the PCB to implement them but they did not included the modules. As they have thought to implement them, we will review their implementation but we will not be including them on the firmware section (Section 2.3).

2.1.1.1.1 Microprocessor

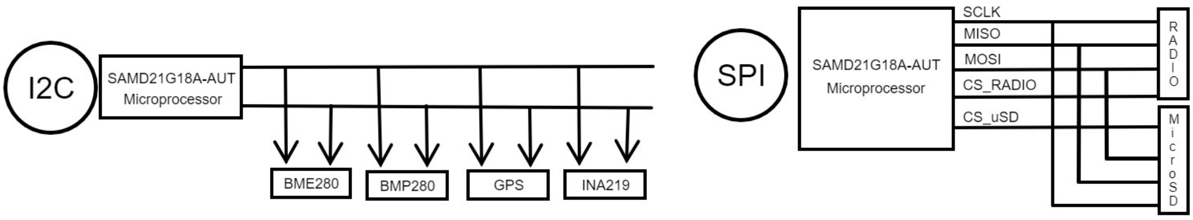
The microprocessor they have used is a low-power 32-bit Cortex-M0 MCU with advanced analog and Pulse Width Modulation (PWM). This is also the microprocessor used by the Arduino Zero board which is useful since it will allow us to program with Arduino IDE without any problem. It is also important to know that this microprocessor has a single core running up to 48 MHz with a Single-cycle hardware multiplier that means that it takes only one clock cycle to perform the operation. Also it has 48 programmable pins that can implement the peripherals listed below [28]:

- 256 kB of flash memory and 32 kB of SRAM memory
- 12 channel Direct Memory Access Controller (DMAC)
- 12 channel Event System
- Up to five 16-bit Timer/Counters (TC)
- Up to four 24-bit Timer/Counters for control (TCC)
- 32-bit Real Time Counter (RTC) with clock/calendar function
- Watchdog Timer (WDT)
- CRC-32 generator
- One full-speed (12 Mbps) Universal Serial Bus (USB) 2.0 interface
- Up to six serial Communication Interfaces (SERCOM),each configurable to operate as either: USART (full-duplex and single-wire half-duplex configuration), I2C (up to 3.4 MHz), SPI, LIN slave
- One two-Channel I2S interface
- One 12-bit, 350 ksps ADC with up to 20 channels
- 10-bit, 350 kbps DAC
- Up to four Analog Comparators (AC) with window compare function
- Up to 256-channel capacitive touch and proximity sensing PTC

Figure 2.7 – *SAMD21g18a-aut microprocessor [29]*

2

Once we have studied the microprocessor datasheet, we can study the connection between the different modules. Although this connections are shown in Section B.1, they are schematized for a greater comfort in Figure 2.8.



(a) *I2C interfaces* **(b)** *SPI interfaces*

Figure 2.8 – *Modules Connection on the Open CanSat kit's MainBoard*

2.1.1.1.2 Temperature, Humidity and Presure Sensor

As we have mentioned on Subsection 1.2.1, the **Primary Mission** the team must develop is to measure the air temperature and the atmospheric pressure after launching the CanSat and during its descent. In order to do that, this kit contains two modules for developing this mission.

- **BMP280 :** Is a digital pressure sensor that can measure not only the pressure of the air but also the temperature with a resolution of 0.16 Pa and 0.01 ºC respectively with a low current consumption of 2.7 μA. The measurement rate of this module is up to 157 Hz. [22]

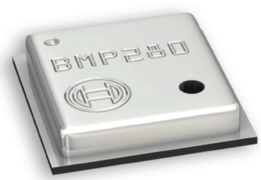


Figure 2.9 – *BMP280 sensor[22]*

- **BME280 :** Is a digital humidity, pressure and temperature sensor that can measure the temperature, humidity and pressure of the air with a resolution of 0.01 ºC, 0.008% RH and 0.18 Pa respectively with a low-current consumption of 3.6 μA.[21]

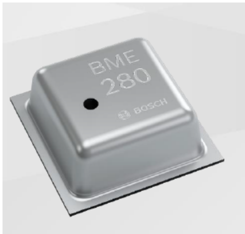


Figure 2.10 – ***BME280*** *sensor [21]*

As we have already mentioned, even though they have implemented a footprint for both sensor, in reality they have only included the BME280 sensor as it can measure not only the temperature and pressure that the BMPE280 sensor measures, but also the air's humidity.

Both sensor can be connected with an I2C or SPI communication but they have chosen I2C communication for both sensor as shown in Figure 2.8 since it is cheaper to implement as it only needs two wires. Beside that, this protocol is also less susceptible to noise even though it is slower than SPI. Other advantages that this protocol can ensure us over SPI are listed on [79].

The circuit implemented for the BMP280 is the one suggested on the datasheet (Figure 2.11) [22] which requires two capacitor of at least 100 nF on the *VCC pin* and a connection between *CSB pin* and the *VCC pin* for selecting the I2C communication.

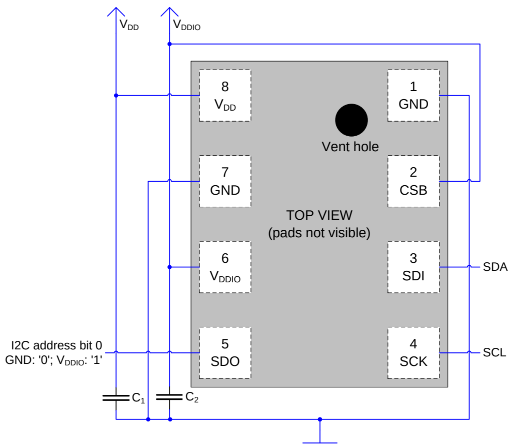


Figure 2.11 – ***BME280*** *sensor application circuit [21]*

The circuit implemented for the BME280 is not exactly as suggested on the datasheet

(Figure 2.12) [21] because, even though they have implemented the two required 100 nF capacitors on the *VCC pin*, and the *CSB pin* connection to the *VCC pin* for selecting the I2C communication, they have not implemented the two pull up resistors required on the *SDA pin* and *SCL pin*. In fact, if the I2C communication had not been selected through the *CSB pin* we would think that they wanted to implement a SPI communication as this connection diagram correspond to that suggestion.

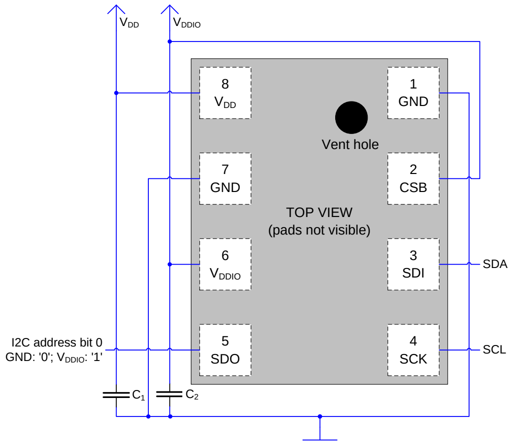


Figure 2.12 – ***BMP280** sensor application circuit [22]*

It is interesting to remark that this sensors are very suitable for this application not only for the electrical advantages they offer, but also for their dimension and price.

2.1.1.1.3 MicroSD Card Socket

Another important module for achieving the **Primary Mission** is the microSD card module as it is required to save the data once per second on a SD card for obtaining the data if the radio communication fails.

The connector they have chosen is a microSD one without a card detection switch. This means that we can not know whether there is a card connected or not. It is remarkable that this connector can handle a maximum voltage of 10 V AC and 0.5 A per contact.[63]

As for the circuit they have suggested is a microSD circuit of 8 pins in which there is a SPI communication. They have also included a decoupling capacitor of 100 nF on the *VCC pin* to prevent the possible interference of the other modules.

Figure 2.13 – 502774 *microSD connector[63]*

2.1.1.1.4 Radio Transmitter Module

The last thing needed for achieving the **Primary Mission** successfully is a radio transmitter. This module is in charge of sending the data collected by the sensors to the ground station.

The radio module they have used is the RFM69W RF transceiver capable of operating over a wide frequency range including 433,868 European and 915 MHz North-America liscence-free ISM frequency bands. This module only requires a 43 mA in transmitter mode and can implement FSK,GFSK, MSK and OOK modulations. Also this modules includes a link budget in excess of 140 dB and a 66 byte TX/RX FIFO.[16]

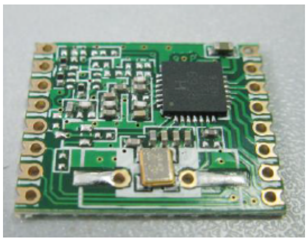


Figure 2.14 – *RFM69W RF radio [16]*

The circuit they have implemented is the one suggested on the datasheet [16]. They are interfacing the module with an SPI communication as it is the only interface the module implements. They have also used two capacitors, a bypass capacitor of 10 μF and a decoupling capacitor of 100 nF for reducing all the possible interferences and noise.

For the antenna they have chosen a 17 cm electronic wire to act as a monopole antenna. As on the CanSat the wire must face the receiver, it should be placed so that the board acts as the ground plane. As a monopole antenna is a $\frac{\lambda}{4}$ antenna, it is needed a electronic wire length such that:

$$f = \frac{c}{\frac{\lambda}{4}} \tag{2.1.1}$$

$$\frac{\lambda}{4} = \frac{c}{f} = \frac{310^8 \text{ m/s}}{433 \text{ MHz}} \approx 17.32 \text{ cm} \tag{2.1.2}$$

2.1.1.1.5 GPS Receiver Module

As well as preparing this kit to achieve the **Primary Mission** they have equipped the device for achieving a simple **Secondary Mission**. For that reason they have included a GPS module. The task for this device is to locate the CanSat in case the landing is not where expected. Additionally, this module can provide us with some relevant information such as the time and date, velocity, altitude and the reached satellites.

It is really important to not only save the GPS plot in the SD but also to transmit this information through telemetry. Otherwise, if the CanSat does not land where expected, it will be lost forever.

The GPS module they have used is the MAX8C model from uBlox. This module is able to receive GPS, GLONASS, QZSS and SBAS signals. In particular, this model is optimized for cost sensitive applications with lowest power. The matching antenna must be a 50 dB antenna and the current consumption of this device is only 17 mA in acquisition and 16 mA in tracking. [61]

Figure 2.15 – *MAX8C uBLOX GPS [61]*

As for the GPS antenna they have used is a active patch antenna without cable as is the better solution for receiving a high strength in a small form factor. This type of antennas consist of a patch of metal directly above a ground plane. The patch is mounted on a PCB. Under this PCB is the LNA which amplifies the GPS signal before being transmitted through the pad. It is important to note that the larger the ground plane, the higher gain the antenna will have (it will take less time to achieve the satellite) [78][26].

Another important factor of the antenna is that the operating frequency is 1575 GHz, as it is a frequency configured for all GPS. It is also circularly polarized (which means that if we do not orientate the antenna up, we will not receive any signal from the satellites) [26].

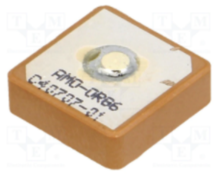


Figure 2.16 – *ORG12-4T-GNSS Patch Antenna*[13]

The circuit implemented for this module is as suggested on the datasheet [61]. As we already mentioned on Figure 2.8, the communication between the microprocessor and this module is through I2C. On the *VCC pin* it has a bypass capacitor of 10 μF and a decoupling capacitor of 100 nF for reducing all the possible interferences and noise. As the patch antenna is a **passive antenna**, it does not need to be powered therefore, the pin designated for powering the antenna by the module is left floating.

2

2.1.1.1.6 LEDs

They have included 6 SMD LEDs to indicate several events as listed:

LED NAME	COLOR	USE
LED1	BLUE	3.3 V power indicator
LED2	RED	A LED for indicating when the module MCP73831 is charging the battery
LED3	RED/BLUE	LED programming indicator. It blinks when a program is uploading
LED4	RED	Blinking when transmitting from the microprocessor
LED5	GREEN	Blinking when receiving information on the microprocessor
LED6	YELLOW	LED for personal use. It can be programmed for the use that the user prefers

Table 2.2 – *LED use and type information*

***LED*1** is implemented as a simple LED with a 10k resistor powered with 3.3 V and with the LED cathode to the ground. Therefore, as long as there is 3.3 V on the power line, the led will be ON.

***LED*2, *LED*4** and ***LED*5** are implemented with a simple circuit as well. It is implemented as a led with a 470R Ω SMD resistor powered with 3.3 V but connecting the LED cathode to an microprocessor pin. This way, when the pin is HIGH, the LED will be OFF but when the pin is LOW, the LED will be ON.

***LED*3** is a LED that implements two colors in one SMD LED. Depending on the state of the pin connected to microprocessor is LOW or HIGH, it will be on one color or the other. Each LED comes with a 470R Ω SMD resistor and there is a 4.7k SMD resistor between the anode of the upper LED and the cathode of the bottom LED.

The last LED, ***LED***6 , is a LED that includes a circuit with a FET. The anode of the LED is connected to a 470R Ω SMD and the cathode to the drain of a power MOSFET. On the gate of the MOSFET there is a high SMD resistor (1 M) , and the source is connected to ground. This structure is implemented for achieving:

2

- Faster switching speeds without delays
- "Controling" higher currents than what the GPIO can provide.

2.1.1.1.7 Current \Power Monitor

The INA219 is a current shunt and power monitor with I2C or SMBUS-compatible interface. The device monitors both shunt voltage drop and bus supply voltage, with programmable conversion times. This module is very effective to bring students closer to the power management control [45].

With a power supply range of 3 V to 5.5 V it can measure a shunt voltage drop from − 26 V to + 26 V by only consuming 5 mA.

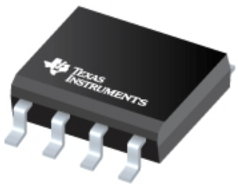


Figure 2.17 – ***INA***219 *Current \Power Monitor [25]*

As for the circuit they have used for connecting this device is the one suggested by the datasheet [45] (Figure 2.18) except that the had not added the pull-up resistor for the I2C interface. They had not add them just because the track connecting this module to the microprocessor is short enough. Also they have included a schottky diode with the purpose of protecting the input power because this devices are more likely to avoid aberrant behavior as well as damage (they also have high ESR as well).

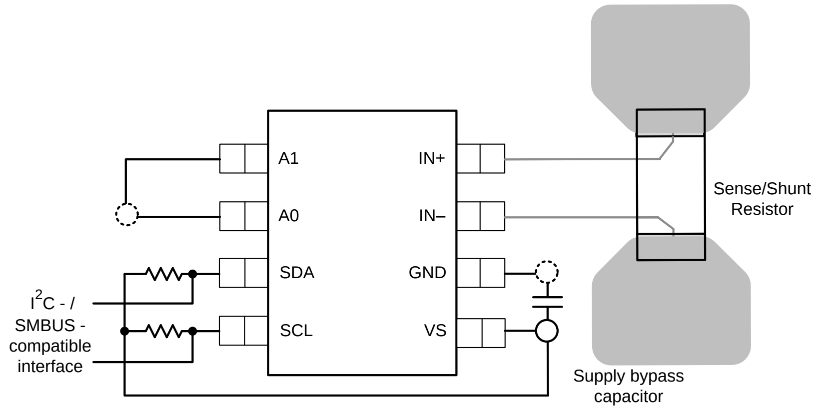


Figure 2.18 – *INA*219 *Current \Power Monitor suggested circuit*[45]

2

On the power supply pin they have added a 100 nF decoupling capacitor. As for the shunt resistor they have used is a 100 Ω 1206 SMD resistor. With a Programmable Gain Amplifier (PGA) of 8 bits they can estimate a voltage range through the shunt resistor of ± 320 mV sense and a resolution of 0.79 mA. The maximum current through the shunt resistor that can be measure is 3.2 A as shown in Equation 2.1.3.

$$I_{max} = \frac{\mathrm{V_{shunt}}}{\mathrm{R_{shunt}}} = \frac{320\ \mathrm{mV}}{100\ \Omega} \tag{2.1.3}$$
$$I_{max} = 3.2\ \mathrm{A}$$

2.1.1.1.8 Hardware Reset

For resetting the device they have included a simple circuit in order to reset the microprocessor. This circuit includes a voltage divider with a switch and a capacitor. With this configuration, showed in Section B.1, when the switch is not pressed, the tension on the microprocessor pin is 3.3 V but when the switch is pressed, this tension is equal to:

$$V_{RST} = 3.3 \cdot \frac{100}{100\ +\ 100^3}\ \mathrm{V} \approx 0.0327\ \mathrm{V} \tag{2.1.4}$$

As the voltage threshold of the microprocessor is 1.0 V and the reset is active low, the condition is met [28]. Therefore, when the switch is pressed, the microprocessor will reset.

2

2.1.1.1.9 USB + Power Management

On this section we will include not only the circuit they have implemented for turning the 3.7 V battery tension to 3.3 V or 5 V needed for the sensors but also how they connect the board to the computer, how they have thought of charging the battery or how they program the different sensors.

- **Programming Pins**: A programming and debugging functionality implemented on the microprocessor is key since as we are programming directly on the microprocessor, if we somehow damage the bootloader file implemented, we will lose the communication between the computer and the board via USB. With this interface, we will be able to restore the bootloader and to communicate again through the USB port. The process for restoring the bootloader will be described on Section D.1. For interfacing the microprocessor with the J-Link device, the device should be connected using the Serial Wire Debug (SWD) interface. The other 3-pins are for *VCC*, *GND* and *RESET*. [28]

- **USB Circuit**: For the micro USB (Figure 2.20) they have used a type-B connector. The circuit used for this device is very simple. They only have added a 10 μH inductor on the *VCC* pin to avoid abrupt current changes followed by a schottky diode for protecting the input power. They have interfaced this module through the D- and D+ pin to the ESD protector.

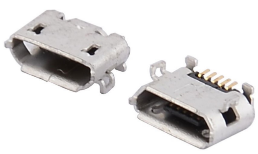


Figure 2.19 – ***Type-B micro USB** connector*[48]

- **ESD Protector**: In order to protect the USB circuit from a ESD to prevent a malfunction or breakdown of the device, they have added a USBLC6 − 2SC6Y ESD protector. This device is dedicated to ESD protection of high speed interfaces with a fast response time. [23]

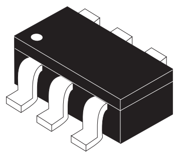


Figure 2.20 – ***ESD Protector*** *[23]*

For interfacing this module they have only added a 5 V power input from the USB.

On the input line they have connected the D+ and D- signals from the USB data lines and on the output, they have connected those lines to the microprocessor.

- **LDO Regulator**: The MCP1826S (Figure 2.21) is a 1000 mA LDO linear regulator that provides high-current and low-output voltages. This module is able to fix the output voltage from 0.8 V to 5.0 V according to the users needs. [17]

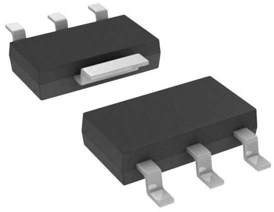


Figure 2.21 – *LDO Regulator* [18]

As for the circuit they have used, on the input lines they have added a bypass capacitor and a decoupling capacitor on the 3.7 V power pin. On the output line they have added a PTC of 500 mA to prevent any overcurrent to damaged the device followed by a switch to act as the ON\OFF button. As indicated on the datasheet ([17]), on the output line it is needed a 1.0 μF so they have added a 100 nF decoupling capacitor and a 10 μ F bypass capacitor.

- **Li-Polymer Charge Management Controller**: The MCP73831 devices are highly advanced linear charge management controllers. With a high accuracy preset voltage regulation of $\pm$ 0.75 % they can regulate the voltage regulation among 4 options (4.20 V, 4.25 V, 4.40 V, 4.50 V) as well as programming the charge current from 15 mA to 500 mA. [14]

Figure 2.22 – ***Li-Polymer Charge Management Controller*** *Module* [19]

As for the circuit they have implemented for this module is the one suggested by the datasheet [14] as shown in Figure 2.23. It is remarkable that even though on the datasheet it is suggested to add a 4.7 μF ceramic capacitor on the input line as well as on the output line, they have replaced both ceramic capacitors with two electrolytic capacitors (one on the input line and an other on the output line) with the same capacitance.

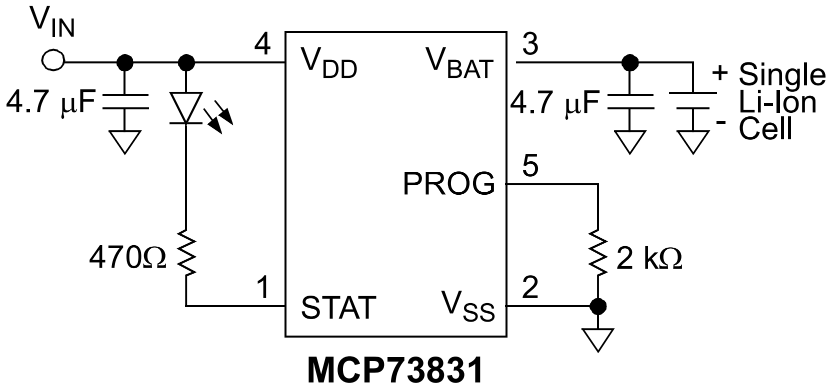


Figure 2.23 – ***Li-Polymer Charge Management Controller*** *Circuit [14]*

We must emphasize that although the USB module is powered by the computer itself with a voltage of 5 V, this is simply a step element to be able to establish the communication between the microprocessor and the computer. The communication lines *D+* and *D-* from the USB are directly connected to the microprocessor therefore, in order to correctly communicate with the computer, we must power the microprocessor. For powering the microprocessor we will only need to connect the main bus connector of the power board to the main bus connector of this board.

2.1.1.2 Uni Board Hardware

As we have already mentioned on Subsection 1.2.3, this board (Figure 2.5) is used for adding extra modules to implement a more complex mission. For that reason, this board has only a pin header for transmitting the signals, from the power board and the main board to this board, and a LED to indicate when the board is connected to the power board (a SMD LED connected to a 10k resistor powered with 3.3 and with a GND connection on the cathode).

For adding new modules, the board is filled with multiples PTH with a pitch of 2.54 mm. In this way, we can add SMT modules as well as SMD.

2.1.1.3 Power Board Hardware

This board, as its name suggests, is the one used for powering the CanSat (transmitter). For achieving this objective, this board includes:

- **Lithium Polymeter Battery**: A 3.7 V and 550 mAh battery remarkable for its small size.

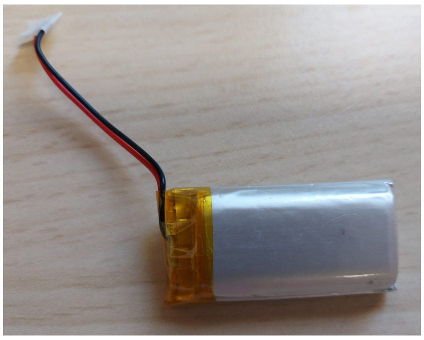


Figure 2.24 – 3.7 *V* - 500 *mA Lithium Polymeter Battery*

- **Blue LED**: A simple LED with a 10 k resistor for indicating when the battery is charged.
- **Bypass capacitor**
- **Decoupling capacitor**
- **Thermistor (PTC)**: Used for protecting against an overcurrent higher than 500 mA.

2.1.1.4 Base Board Hardware

As for the modules implemented on this board, the receiver board, we can identify some of the modules we observed on the main Board such as the radio, the GPS, the microprocessor, resistors, LEDs, etc. If we proceed to study the Eagle's SCH project as we did on subsubsection 2.1.1.1, we are able to gather the following information about the modules extracted from their datasheet:

Module	Model	Information
Microprocessor	SAMD21G18A-AUT	Device's brain. Microprocessor from *SAMD 2X Series Microcontrollers Family* in charged of executing all the module's programs
GPS	uBLOX-MAX8C	The GPS is in charged of locating the device in case the landing goes wrong. It can also measure some interesting parameters as the device velocity
GPS Antenna Connector	SMA Female Adapter	A 50 Ω impedance coaxial RF connector

Continued on next page

2

Module	Model	Information
Radio	RFM69W	Transmitter module needed for sending the data to the ***Base Station*** with a 433 MHz frequency
Radio Antenna Connector	SMA Female Adapter	A 50 Ω impedance coaxial RF connector
MicroSD socket	MX-502774-0891	A microSD socket for connecting an SD to save the measurements. This module is needed for obtaining the measurements in case the communication between the *transmitter* and *receiver* is down
Bluetooth Module	HC−05 ZS−040	A module for establishing a transparent wireless serial connection. Data can be sent to any device who implements a bluetooth communication
Display + SD module	1.8″ TFT SPI 128x160 V1.1	A thin-film-transistor (TFT) liquid-crystal display (LCD) with an SD socket for displaying the data received through radio on the base board.
Programming Pins	SAMTEC-A-TSM	A 5x1 Join Test Action Group (JTAG) port with a clock + single bi-directional data pin, providing all the normal JTAG debug and test functionality plus real-time access to system memory without halting the processor or requiring any target resident code)
USB Connector	type-B microUSB	A microUSB connector for establishing the communication with the computer through the serial port and for charging the module
Electrostatic Discharge ESD Protector	USBLC6-2SC6Y	An ESD protector to prevent the possible surges derived from the human's board manipulation

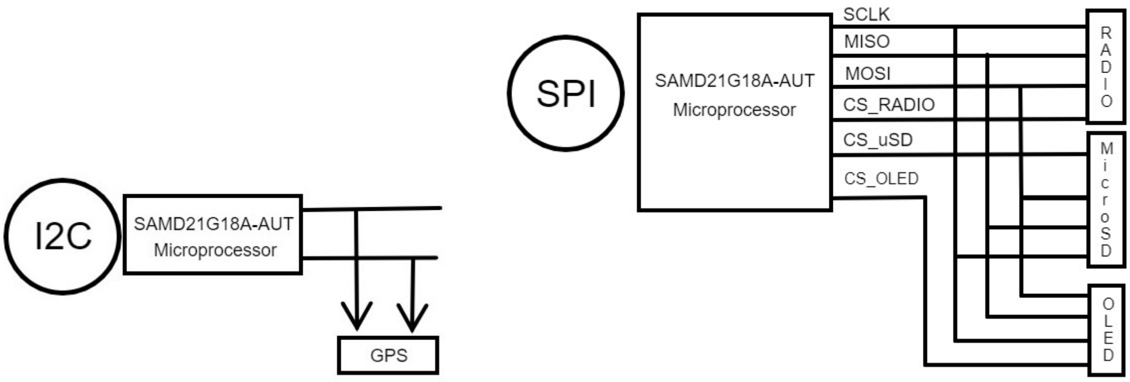
Continued on next page

Module	Model	Information
Battery Charger	MCP73831	A module for charging the device through the USB. From this module we will also take the +5 V voltage supply
LDO Regulator	MCP1826S	A module used for converting the battery's 3.7 V tension into 3.3 V tension used to power the different modules
Main Bus Connector	13x2 female pin header	A pin header to connect this board to the **Power Board** and to the **Uni Board**

Table 2.3 – *Main Board Modules Description*

2.1.1.4.1 Microprocessor

As the microprocessor they have used is the same used in the Main Board, we will refer to paragraph 2.1.1.1.1 to see its characteristics. Once we know all the functions that this microprocessor has extracted from the datasheet [28], we can study the connections between the different modules. Although this connections are shown in Section B.1, they are schematized for a great confort in Figure 2.25.



(a) I2C *interface's modules* **(b)** SPI *interface's modules*

Figure 2.25 – *Modules Connection on the Open CanSAT kit's BaseBoard*

2.1.1.4.2 MicroSD Card Socket

The data received by radio will need to be processed later to study what has happened during the flight and if this data received is consistent,therefore we need some method that allows us to save everything that is received by radio. As the internal memory of the microprocessor is not large enough to store all this data, we will need a SD that allow us to store all the data for further processing.

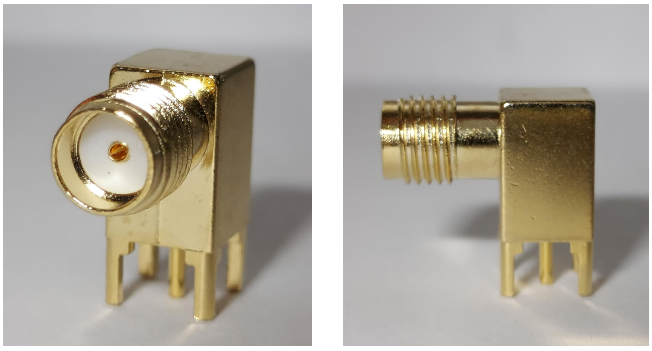
To do so, they have implemented a SD socket identical to the one used in the Main Board. The module and the connection is the same as the one on the Main Board, so we will refer to paragraph 2.1.1.1.3 to better understand the circuit and characteristics.

2.1.1.4.3 Radio Receiver Module

For achieving the **Primary Mission** successfully we need to received all the data measured with the temperature and pressure sensor via radio, hence this module is key on the Base Board.

The module they have chosen to perform this function is the one chosen on paragraph 2.1.1.1.4. As they have chosen the same module, they have implemented the same circuit.

However it is remarkable that for the antenna they have implemented a SMA female adapter (Figure 2.26) so that the user could chose their own antenna (Yagui, OmniDirectional, etc).



(a) *I2C interface's modules* **(b)** *SPI interface's modules*

Figure 2.26 – *SMA Female Adapter Right Angle with* 50 Ω *Impedance*

The only requirements that this antenna must met is that it must be a 433 MHz frequency antenna so that it matches the frequency of the transmitter module. Also, it needs to have a 50 Ω matching impedance.

2.1.1.4.4 GPS Receiver Module

Seeing that on the Main Board they included a GPS module for achieving a **Secondary Mission** they have decided to add also a GPS module on the Base Board. This module could be used for comparing the coordinates received from the Main Board to the ones received on this board to keep track of how far the CanSat is. For that reason, they have decided to choose the same module used on this main board so we will be referring to paragraph 2.1.1.1.5 for studying it characteristics and the circuit implemented.

2.1.1.4.5 LEDs

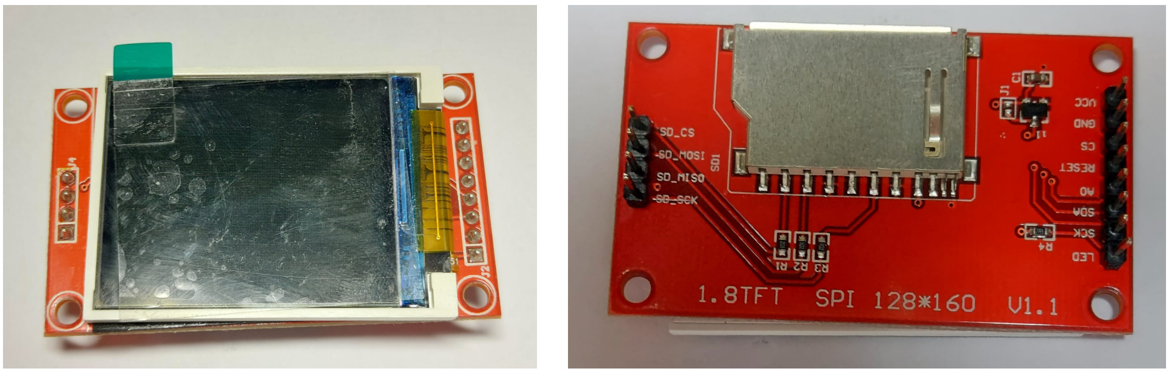
To be consistent with the LEDs indicators on the Main Board, they have chosed to include the same set of LEDs with their respective circuits. Therefore, we will be turning to paragraph 2.1.1.1.6 for studying them.

It should be noted that even though on the SCH they have not included the LED for indicating when the battery is being charged (LED2), they have implemented it on the board.

2.1.1.4.6 TFT LCD + SD

The 1.8 TFT LCD (Figure 2.27) is a colorful display with 128x160 color pixels. This module also includes a SD socket on the bottom so that images can be loaded to the display. Both functions are interfaced via SPI.

This module is in reality a PCB that includes all the indispensable modules to make the SD and the display work.



(a) *Front View* **(b)** *Bottom View*

Figure 2.27 – *TFT LCD + SD Device*

As for the SD module included on this device is a common 12-pin SD. This module is interface via SPI and needs three 1 kΩ resistors on the *CS* or *SS* pin, *MOSI* pin and on the *SCLK* pin.

For the display they have used a ST7735S single-chip controller for 262K-color, graphic type TFT-LCD which consists of 396 source line and 162 gate line driving circuits [20]. As the connection of this display is made through a flex cable, the 1.8 TFT LCD *V*1.1

intends to make the connection to any microprocessor much simpler. In consequence, the connection between the module and the microprocessor is made through the SPI interface. On the on-board module it includes a 7.5 resistor on the *SCLK* pin and a DC-DC converter to power down the *VCC* line to 3.3 V in case the module is powered with 5 V.

2

2.1.1.4.7 Bluetooth Module

For achieving a second mission they have included a Bluetooth module. This module is a Serial port module fully qualified with Bluetooth $V2.0 + EDR$ (Enhanced Data Rate) 3 Mbps Modulation with complete 2.4 GHz radio transceiver and baseband. It uses Certificate Signing Request (CSR) Bluecore 04-External single chip Bluetooth system with CMOS technology and with Adaptive Frequency Hoping Feature (AFH). [24]

It is remarkable that it has a low power operation from 1.8 V to 3.6 V and can be interfaced with UART and programmable baud rate. Therefore, the circuit they have implemented is very simple. They have only added a 3.3 V power on the *VCC* pin , the UART interface crossing the pins *RX* , *TX* and the *GND* to mass.

Figure 2.28 – ***HC*-05 *Bluetooth*** *to Serial Port Module [12]*

2.1.1.4.8 Hardware Reset

Since the microprocessor they have used on the Base Board is the same as the one used on the Main Board, the process for resetting the module is the same as the one studied on paragraph 2.1.1.1.8.

2.1.1.4.9 USB + Power Management

By last, we should mention the circuits implemented for powering the board. As they have used the same system as the one explained on paragraph 2.1.1.1.9 we will remit to that paragraph for more information regarding the modules or the circuits implemented. However we should mention that although on their SCH they have not included the module for charging the battery (MCP73831), we have noted that it is implemented on the board. In fact, if this module were not implemented, it would be pointless to place the battery since once it is discharged for the first time, it can no longer be recharged and it should not make sense to have it.

2.2 Power Budget

Once we are in possession of the information regarding all the modules that the Open CanSat boards implement, we can elaborate their power budget. This power budget is a way to estimate whether the battery election is optimal or not taking into account the usage time of the entire device. Given that we have two different boards, the transmitter and the receiver, we will be implementing a power budget for both of them. The optimal tool to carry out this budget is any spreadsheet.

The first thing we have to estimate when doing a power budget is the modules consumption, for that reason we have extracted from their datasheet all the modules consumption and calculated its power consumption.

Component	Module	Number/PCB	V_{IN} (V)	I_{IN_TYP} (mA)	I_{IN_MAX} (mA)	P_{IN_TYP} (W)	P_{IN_MAX} (W)
uProcessor	SAMD21G18A-AUT	1	3,3	276	390	0,91	1,29
microSD	SanDisk standard uSD	1	3,3	30	100	0,10	0,33
GPS	uBlox-MAX8C	1	3,3	17	67	0,06	0,22
	Ceramic Antenna	1	3,3	4,5	10	0,01	0,03
	Exterior Antenna	1	3,3	4,5	10	0,01	0,03
Radio Transmitter	RFM69W	1	3,3	33	130	0,11	0,43
Radio Receiver	RFM69W	1	3,3	16	20	0,05	0,07
Pressure				2,80E-03	4,20E-03	9,24E-06	1,39E-05
Humidity Sensor	BME280	1	3,3	1,80E-03	2,80E-03	5,94E-06	9,24E-06
Temperature				1,00E-03	2,00E-03	3,30E-06	3,60E-09
Pressure and Temp. Sensor	BMP280	1	3,3	2,80E-03	4,20E-03	0,01	0,01
Display	LCD 128x160	1	3,3	80	100	0,26	0,33
Bluetoth	MOD2	1	3,3	30	50	0,10	0,17
Red LED	0805 SMD	2	3,3	1,53	10,58	0,01	0,03
Blue LED	806 SMD	2	3,3	1,96	1,66	0,01	0,01
GreenLED	807 SMD	1	3,3	1,00	1,13	0,00	0,00
osram LED	2 in 1 Leds	1	3,3	1,00	5,29	0,00	0,017
Yellow LED	809 SMD	1	3,3	0,76	1,13	2,49E-03	3,73E-03
DC Current Measurement	INA219A	1	3,3	0,7	1	2,31E-03	0,0033

Figure 2.29 – *Open CanSat modules consumption*

Once we have estimated each module consumption, we can determine the current and power needed for both boards. We also can present the battery characteristics.

	TYP	MAX
V_{SOURCE} (V)	3,75	4.2
CAPACITY (mAh)	550	570
OUTPUT CURRENT (A)	1	1,1
TOTAL POWER MAIN BOARD(W)	1,21	2,37
TOTAL POWER BASE BOARD(W)	1,52	2,49
CURRENT NEEDED MAIN BOARD (A)	0,37	0,72
CURRENT NEEDED BASE BOARD (A)	0,46	0,75
SOURCE SPECS	3.7 V / 3.3 / 5 V	3.7 V / 3.3 / 5 V

Figure 2.30 – *Open CanSat current and power needs*

Then we can move on to determine the typical (2.31a) and maximum (Figure 2.32) consumption budget according to the buck and boot converters.

TYPICAL CONSUMPTION BUDGET							
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT_MAX} (A)	I_{OUT} (A)	P_{IN} (W)	η
MCP1826 (3.7 V -> 3.3 V)	1	3,7	1,65	0,5	0,5	3,70	45%
MCP73831 (3.7 V -> 5 V)	1	3,7	2,525	0,55	0,505	3,70	68%

(a) *Open CanSat typical DC-DC consumption budget*

MAXIMUM CONSUMPTION BUDGET							
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT_MAX} (A)	I_{OUT} (A)	P_{IN} (W)	η
MCP1826 (3.7 V -> 3.3 V)	1	6	4,125	1,25	1,25	6,60	63%
MCP73831 (3.7 V -> 5 V)	1	6	1,815	0,55	0,55	6,60	28%

(b) *Open CanSat maximum DC-DC consumption budget*

Figure 2.31 – *CH340C chip [36]*

The last thing to do is to calculate the time that the device could be ON given its consumption. To do so, we only have to apply the mathematical calculus presented in Equation 2.2.1.

$$On\_Time = \frac{\text{Current Consumption}}{\text{Battery Capacity}} \tag{2.2.1}$$

Therefore, given the consumption of each board and the batteries capacity, we can estimate that:

	ON TIME (h:min:s)	
	TYP	MAX
MAIN BOARD	0:40:05	1:15:27
POWER BOARD	0:49:58	1:19:06

Figure 2.32 – *Open CanSat ON time*

2.3 Product Firmware

On the Czech Engineers GitHub [3] we can find most of the modules examples for Arduino IDE, but as they did not uploaded an example for the complete CanSat or even an example for some of the modules, we decided to develop our own . To do so, instead of using Arduino IDE which is a more limiting environment, we decided to develop all our programs in Visual Studio Code (VS Code) with PlatformIO extension. It needs to be mentioned that even though Arduino IDE will do the job just fine, it would not be as clear, organized and faster as with PlatformIO in VS Code. Besides, with PlatformIO in VS Code we would have more control over the linking and compiling process.

Once the platform we are going to work on is mentioned, we must go on to mention the philosophy that our firmware will follow. As our microprocessor has only one core with a reduced RAM and is not as powerful as other microprocessors such as the STM32L476RG or the ESP-WROOM-32 we will develop a general purpose operating system also known as **Super Loop** structure.

A super loop is a program structure comprised of a infinite loop, with all the tasks of the system contained in that loop. The initialization routines are performed before we enter the loop function, on the setup, as we only want to initialize the system once. This loop is a variant of the classic *"batch processing"* control flow: read input, calculate some values, write out values. This architecture is used in, for example, computer games [75]. The main inconvenient of this architecture is that when a certain task needs a great amount of time, with this architecture no other task would run until that slow task is done.

Furthermore, as we are going to implement several tasks, instead of implementing all of them on the *main.cpp* we will move them to separate libraries so that our main function is as clear as possible.

2.3.1 CanSat Transmitter firmware

The transmitter has to measure the environment parameters, track the consumption, receive the GPS coordinates and save those variables on the SD in case the radio transmission does not goes as expected. Finally, those parameters need to be send via radio. All the tasks that will be running in our CanSat transmitter are presented in Figure 2.33.

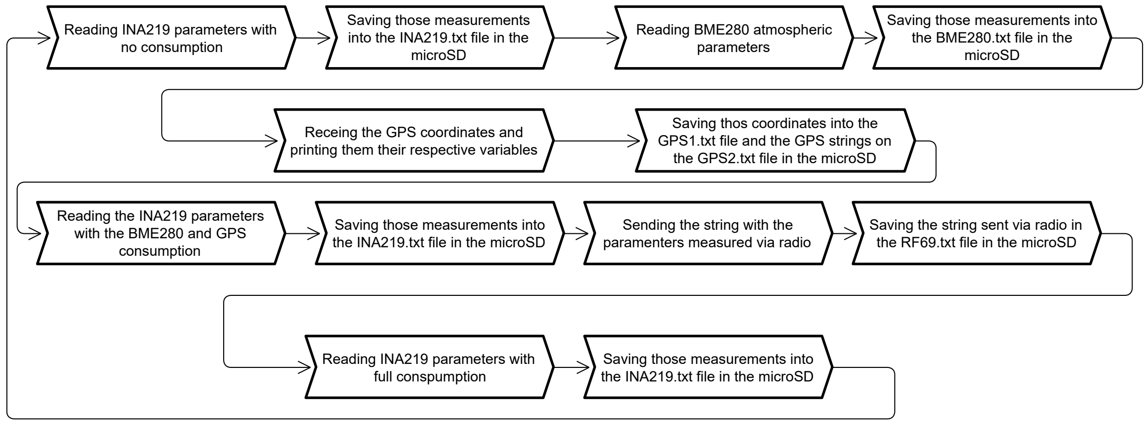


Figure 2.33 – *CanSat transmitter tasks*

To do such tasks, we have developed the firmware exposed on the block diagram of Figure 2.35. Before getting to explain the firmware we have developed, we want to mention that since we want to create a portable firmware that could be used both today an in x years from now, the structure of our project in VS Code is as shown in Figure 2.34.

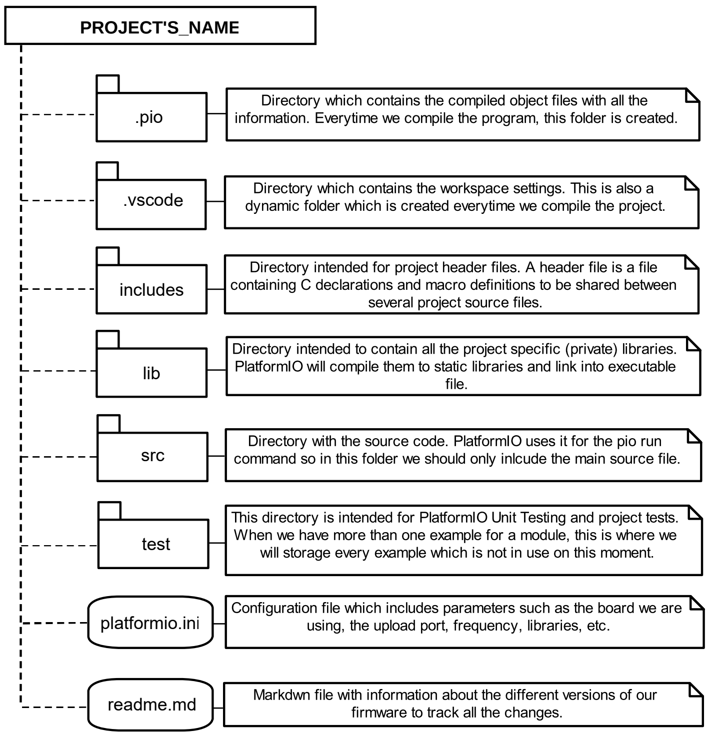


Figure 2.34 – *PlatformIO general project structure*

The libraries and their respective examples have all been downloaded from `github.com` (BME280 lib [64], INA219 lib [68], software RTC lib [73], SD lib [72], GPS lib

[60],RFM69 [27]). The Arduino default libraries are all included on PlatformIO so that we do not have to include them on the lib directory.

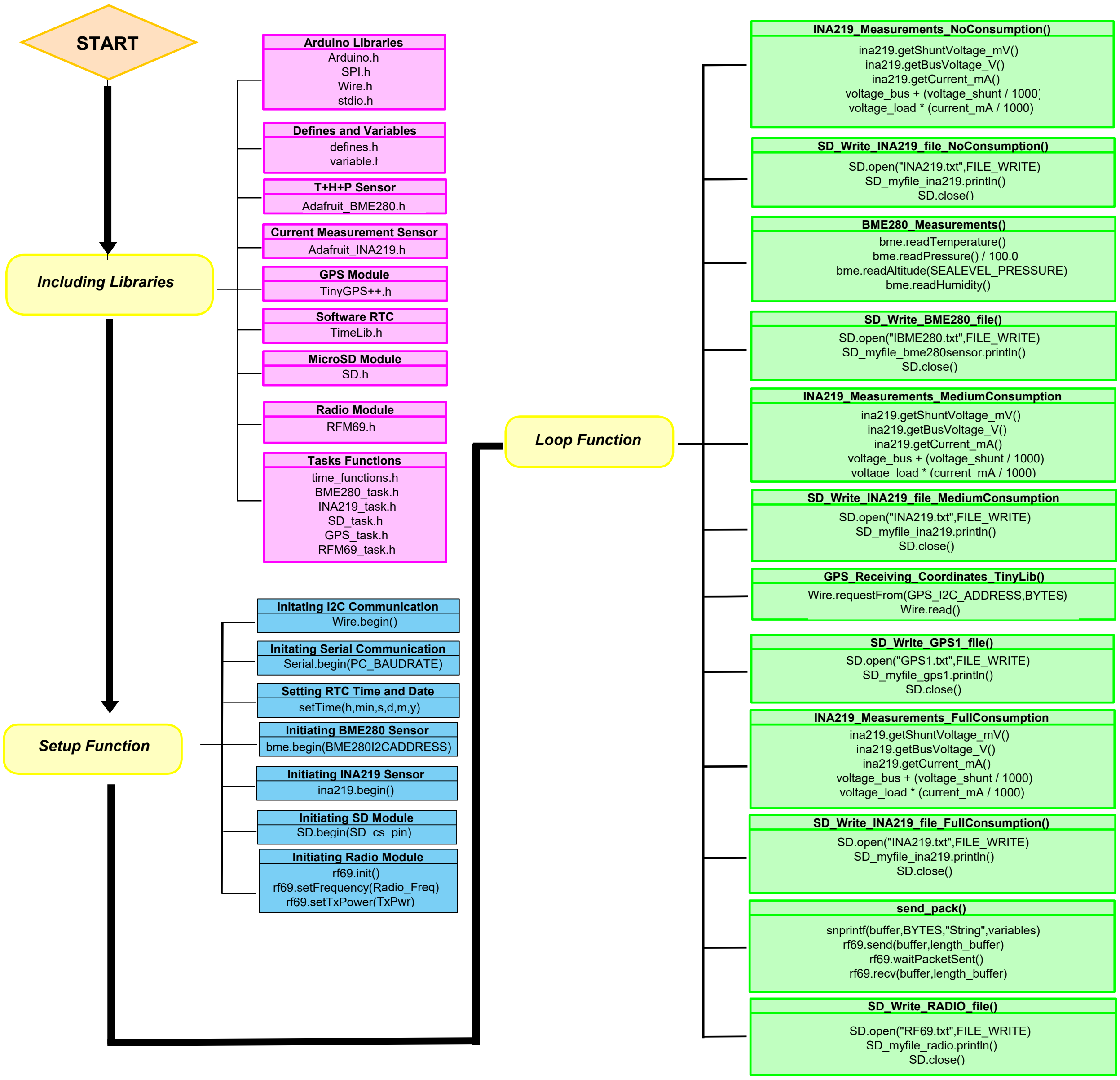


Figure 2.35 – *Block Diagram OpenCanSat Transmitter*

Getting back to the diagram on Figure 2.35, we are going to explain the most relevant parts of the code that we have implemented. First of all we have to remark that, after including all the libraries, we have the two main functions of a program. The **setup function** is a function that will only be executed once, therefore on that function we should include all the modules initializations. Furthermore, on the other function, the **loop function**, is a function that will be running until the user stops it, consequently we should include on that part all the modules tasks. Moreover we should remark that all this libraries are included with the *.h* file. On the *.h* (header) files we include the function's definition whereas on the *.cpp* (source) we will include the real source code which will be compiled. Keeping all of this in mind, we will proceed to explain what we have included in both main functions.

2

2.3.1.0.1 Transmitter Setup Function

As shown on Figure 2.35, on this function we will initialize not only the communication (and its baud rate) between the computer and the board but also the different modules communications with the microprocessor. We have initialized the serial communication as shown in 2.3.1.0.1 setting the PC baud rate with 115200 bits per second in the *defines.h* file.

```
Serial.begin(PC_BAUDRATE)
```

Code 2.1 – *C++ command for initializing the serial communication*

Once we have established the serial communication between the PC and the microprocessor we need to initialize the communication between the microprocessor and the rest of the modules. For the modules interface with I2C we only need to specify their I2C address as this is a point-to-multipoint communication protocol. As for the SPI communication used for the rest of the modules, we only need to specify the CS pin for the module as it is also a point-to-multipoint communication protocol. Given that we are using libraries, before initiating the modules we need to call the module's class. The classes of the different modules are made on the *variables.cpp* file as shown on 2.3.1.0.1.

```
Adafruit_BME280 bme; //Class for using BME280

Adafruit_INA219 ina219(INA219_I2C_ADDRESS); //Class for using INA219 indicating the I2C address

File SD_myfile_bme280sensor; //Defining an object type File
File SD_myfile_gps1; //File for storing the data from the GPS analyzed data
File SD_myfile_gps2;     //File for storing the gps string
File SD_myfile_ina219; //File for storing the ina219 measured data
File SD_myfile_radio; //File for storing the radio strings and parametres

TinyGPSPlus gps;

RH_RF69 rf69(ss_pin, interupt_pin); //create an RFM69 instance
```

Code 2.2 – *Calling the different modules classes*

With the classes initiated, we can check whether the modules are up for communication by initiating the modules as shown in 2.3.1.0.1.

```

Wire.begin(); //Initiating I2C Communication

setTime(h,min,s,d,m,y); //Setting the exact tame and date with the following
```

```
                         //parameters:h,min,sec,day,month,year
  bme.begin(BME280_I2C_ADDRESS); //Initiating BME280 sensor
  ina219.begin(); //Initiating INA219 module
  SD.begin(cs_pin); //Initiating SD module
  rf69.init(); //Initiating Radio module
      rf6.setFrequency(FREQUENCY_RADIO_MODULE); //Setting the 433 MHz frequency for the radio module
  rf69.setTxPower(POWER); //Setting the transmitter power given that the maximum is 20 dB
```

Code 2.3 – *C++ command for initializing the different modules*

Note that depending on the library configuration, we need to specify the I2C address (Table 2.4) on the initialization command as done with the BME280 or on the class calling as done with the INA219. Similarly, with the SPI, some libraries need to specify the CS (Table 2.5) on the module initialization such as the SD module and other on the class calling as done with the radio module. To conclude, we need to remark that the GPS address is not specified either in the class calling or in the module initialization. This address is specified in the GPS task every time we check whether there is some information available to pass from the module to the microprocessor.

MODULE	I2C ADDRESS
INA219	$0x40$
BME280	$0x77$
GPS	$0x42$

Table 2.4 – *I2C addresses of the modules using I2C communication*

MODULE	CS PIN
RFM69	43
microSD	35

Table 2.5 – *CS pin of the modules using SPI communication*

Once all the modules are correctly initiated, we can proceed to start the corresponding tasks on the loop function. Since we have implemented all tasks on the library folder, on the loop function we will only call the function we want to implement. Therefore we will only show the most important commands included on the modules tasks *.cpp* files.

2.3.1.0.2 Transmitter Loop Function

2.3.1.0.2.1 INA219 tasks

For the INA219 we have implemented the code lines exposed on 2.3.1.0.2.1. What this lines really do is to read the corresponding register of the module, multiply or divide the value extracted from the register depending on the parameter we are measuring and saving it into a global variable of our firmware.

```
voltage_shunt = ina219.getShuntVoltage_mV();  //Voltage drop on the shunt resistor
voltage_bus = ina219.getBusVoltage_V();        //Tension that enters the INA219 module
```

```
current_mA = ina219.getCurrent_mA();            //Current running through the shunt resistor
voltage_load = voltage_bus + (voltage_shunt / 1000); //Total tension needed for this module
power_W= voltage_load * (current_mA / 1000);    //total power
```

Code 2.4 – *C++ commands for measuring the INA219 parameters in the INA219_task.cpp [68]*

If we recall from the Figure 2.33, we have to measure this parameters at three different points of our loop function: when there is no consumption, when there is some consumption (BME280 and GPS modules working) and when all the modules are working. The only thing that differs on this three iterations is the value extracted from the register, the rest of the procedures remain the same.

2

2.3.1.0.2.2 BME280 tasks

The code implementation for this module is as shown in 2.3.1.0.2.2. Similarly to the INA219 module, this lines save on a global variable of our firmware the value extracted from the module's register and modified through some math calculation depending of the parameter we are measuring.

```
temperature_bme280 = bme.readTemperature();     //Reading and saving the temperature on its variable
  pressure_bme280 = bme.readPressure() / 100.0F;  //Reading and saving the pressure on its variable. Dividing by
    100.0F we make
                                                  //sure it is a floating point value
  altitude_bme280 = bme.readAltitude(1013.25);    //Reading the altitude saved on the variable with the
   SeaLevelPressure reference
  humidity_bme280 = bme.readHumidity();           //Reading the humidity value and saving it on its variable
```

Code 2.5 – *C++ commands for measuring the BME280 parameters in the BME280_task.cpp [64]*

It should be remarked that this module estimates the altitude in meters based on the pressure at the sea level. We have assumed that the sea level pressure is the generic one (1013.25 hPa) but if we want to take more accurate measures, we should change this value for the value of the launching place. Besides, the rest of the parameters depends on the altitude, so this is a very important parameter to be taken into account.

2.3.1.0.2.3 GPS tasks

The main code lines that we have to implement for this module are presented on 2.3.1.0.2.3. In reality for this module we only need two code lines. One for requesting the available bytes from the GPS I2C address and one for reading and storing those bytes. The rest of the functions implemented on this file are for saving the corresponding information contained in the string that the microprocessor receives from the GPS into the corresponding global variable of our firmware.

```
Wire.requestFrom(GPS_I2C_ADDRESS, 64); //It is used by the master to ask for bytes to the slave device
                                //0x42 is the address of the GPS (master) and it is requested 64 bytes
  while (Wire.available() > 0) //While there are bytes available
  {
    c = Wire.read(); //Reading the string and storing it into the char c variable
  }
```

Code 2.6 – *C++ commands for receiving the GPS strings (saving the information in the corresponding variable) in the GPS_task.cpp [60]*

2.3.1.0.2.4 Radio tasks

For this module we also need a few code lines. The two main lines needed for this code is one to store all the variables into a string saving it in a buffer and one to send this buffer. This code lines are shown in 2.3.1.0.2.4.

```
snprintf(send_buff_package,100,"%d -> %d,%d,%d,%d,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%d ",packet_number,
  temperature_bme280,pressure_bme280,
        altitude_bme280,humidity_bme280,voltage_shunt,voltage_bus,current_mA,voltage_load,power_W,latitude,
          longitude,knots,num_satelites,
        garbage); //Storing all the listed variables into a string saved on the send_buff_package buffer
    rf69.send((uint8_t *)send_buff_package,length_package_sent); //Passing the string (byte array) to the
      radio to send

    rf69.recv(buffer, &len); //Receiving a string in a buffer to confirm that the package has been received
      successfully
```

Code 2.7 – *C++ commands for sending the package with the variables with the radio module in the RFM69_task.cpp [27]*

It should be noted that when we initiated this module in 2.3.1.0.1 we established the radio parameters. This parameters are not only the frequency and the transmission power. With this initiation we established that the received signal strength indicator (RSSI) threshold would be -114 dBm, the packet would be composed of: 4 bytes preamble, 2 SYNC words, 2 CRC octets and 0 to 60 data bytes and we also have set the speed and modulation as: GFSK modulation, whitening, baud rate = 250 kbs and frequency = 250 kHz.

2.3.1.0.2.5 Software RTC tasks

As we are not in possession of a RTC to give us the exact time of the devices measurements, we need to implement it by software. This implies a lack of precision as we have to indicate the exact time and date and the library calculates the time difference between the library initialization and the time request.

```
hour(); // the time is calculated with the function millis() and the corresponding
        //additions and subtractions to find its difference
minute();
second();
day();
month();
year();
```

Code 2.8 – *C++ commands for requesting the exact time and date in the time_functions.cpp [73]*

2.3.1.0.2.6 SD tasks

Last of all, for printing whatever we want into a *.txt* file into the microSD module we will only need to open the file, printing whatever we need with the corresponding file instance declared on 2.3.1.0.1 and closing the files. To do so, we have presented on 2.3.1.0.2.6 the code lines needed for writing in each file.

```
    SD_myfile_bme280sensor = SD.open("BME280.txt", FILE_WRITE); //opening the files and creating them in case there is no file with that name
    SD_myfile_ina219 = SD.open ("INA219.txt", FILE_WRITE);
    SD_myfile_gps1 = SD.open("GPS1.txt", FILE_WRITE);
    SD_myfile_gps2 = SD.open("GPS2.txt", FILE_WRITE);
    SD_myfile_radio = SD.open ("RF69.txt",FILE_WRITE);

    SD_myfile_bme280sensor.println("CANSAT measured data");
    SD_myfile_ina219.println("Shunt resistor measured electrical parametres");
    SD_myfile_gps1.println("GPS extracted information");
    SD_myfile_gps2.println("CANSAT GPS position");
    SD_myfile_radio.println("CANSAT string sent via Radio");

    SD_myfile_bme280sensor.close(); //Closing the files
    SD_myfile_ina219.close();
    SD_myfile_gps1.close();
    SD_myfile_gps2.close();
    SD_myfile_radio.close();
```

Code 2.9 – *C++ commands for saving information into a microSD file in the SD_task.cpp [72]*

We will refer to the Subsection 2.6.1 to see the modules' proofs.

2.3.2 CanSat Receiver firmware

The receiver only has to measure the GPS coordinates, receive the variables package from the CanSat transmitter and display the GPS coordinates and the variables received from the transmitter on the display. Since the display is controlled with three buttons, we have to structure this firmware as a finite-state machine (FSM) so that every time we press one of the buttons, there is an interrupt that provoke the respective change on the display.

The FSM implemented is the one shown in Figure 2.36.

On this board we will be implementing the same software philosophy as we also want a portable firmware with the main function as clear as possible. For that reason, we will be distributing the project as we already exposed on Figure 2.34.

The modules used in this board are the same we used on the transmitter board therefore the libraries and functions are the same we used on Subsection 2.3.1. For that reason, the libraries used for the GPS module, the microSD module and the radio module are the ones we downloaded from their respective GitHub ([60] [72] [27]). The only module that this board includes that the transmitter did not implement is the LCD and the library used for programming this module is also downloaded from GitHub [62].

To implement the tasks of the receiver FSM exposed in Figure 2.36 we have developed the firmware exposed on the block diagram of Figure 2.37

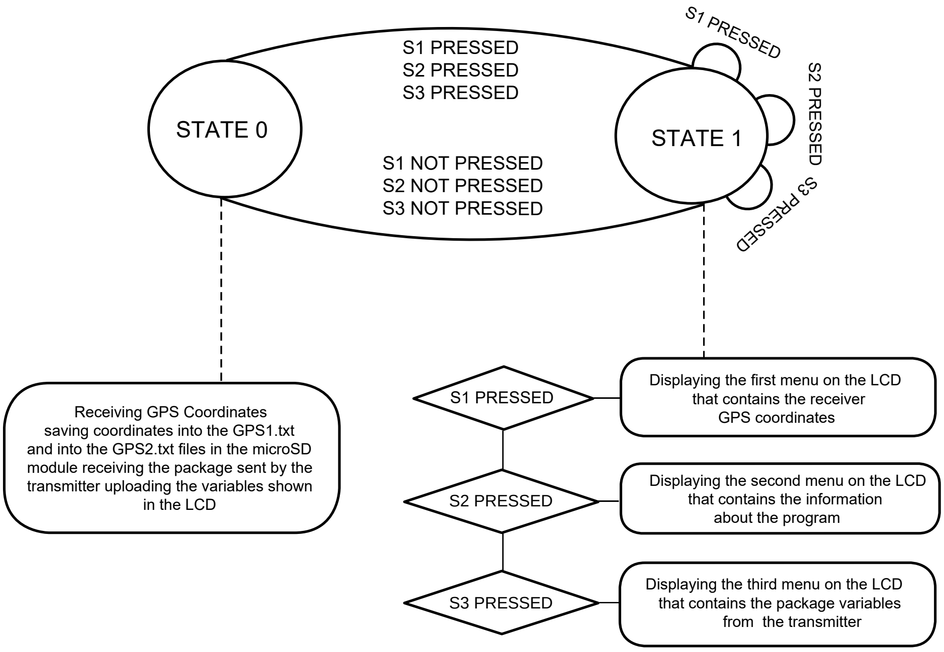


Figure 2.36 – *FSM OpenCanSat Receiver*

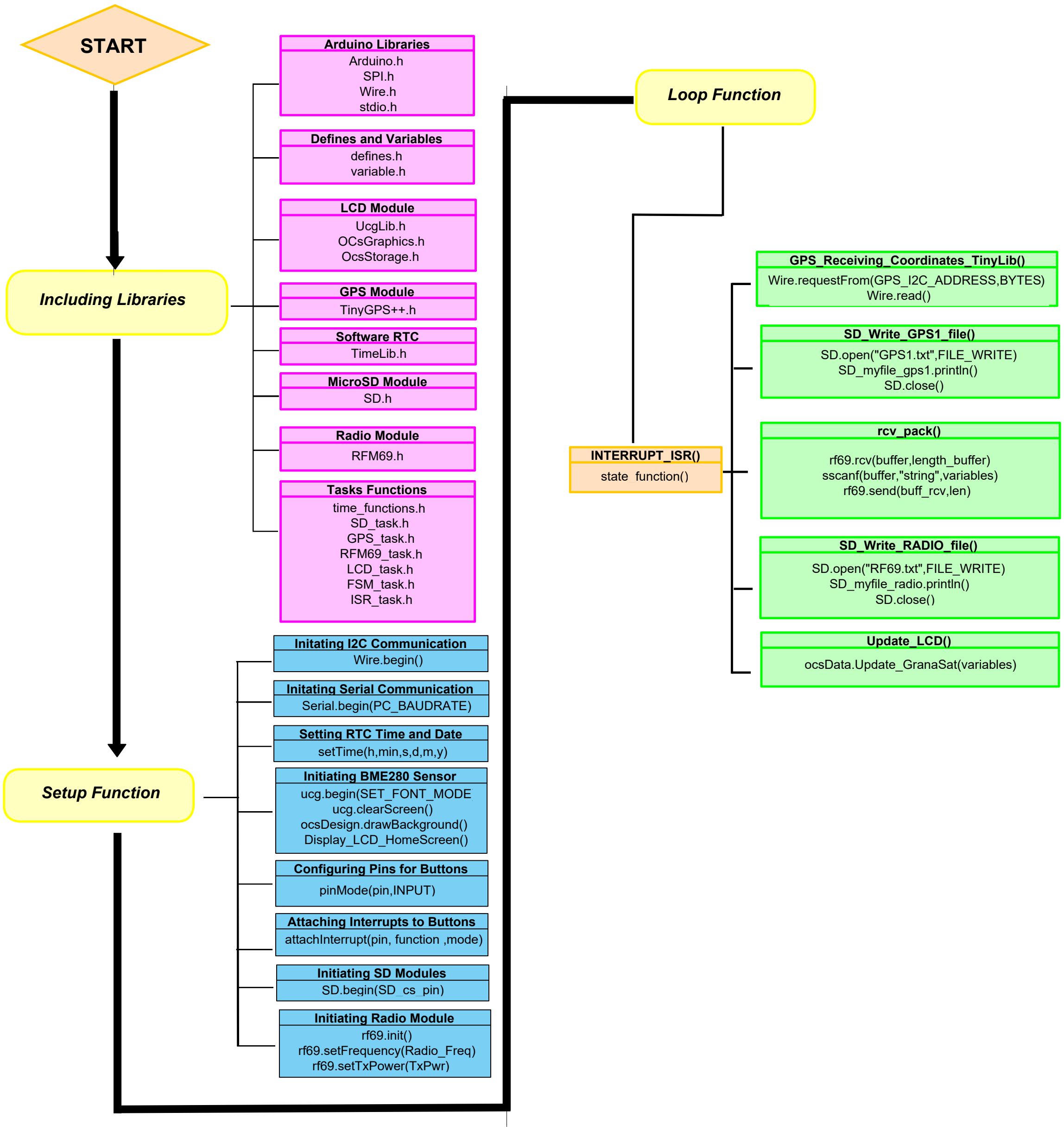


Figure 2.37 – *Block Diagram OpenCanSat Receiver*

Similarly to what we did for the CanSat transmitter, we are going to explain the most relevant parts of the code we have implemented. To do so, we are going to present the different code lines we have included in both main functions, the **setup function** and the **loop function**.

2.3.2.0.1 Receiver Setup Function

The first thing we have done is setting the communication between the microprocessor and the PC as we did on 2.3.1.0.1. The PC baud rate is also 115200 bits per second defined in the *defines.h* file. Then, given that we are using libraries, we need to initiate the module's classes. The code lines implemented to do so are as shown in 2.3.2.0.1. Notice that for the modules used on the transmitter this calls are identical.

```
Ucglib_ST7735_18x128x160_HWSPI ucg(6, 7, -1); //Specifying the cd pin, the cs pin and the reset pin
OcsGraphics ocsDesign(ucg);
OcsStorage ocsData(ocsDesign);

  File SD_myfile_gps1; //File for storing the data from the GPS analyzed data
  File SD_myfile_gps2;     //File for storing the gps string
  File SD_myfile_radio; //File for storing the radio strings and parametres

  TinyGPSPlus gps;

  RH_RF69 rf69(ss_pin, interupt_pin); //create an RFM69 instance
```

Code 2.10 – *Calling the different modules classes*

The last thing to do on this function is to check whether the modules are up for communication and to configure the interrupts we are going to use for the buttons. The code lines we need to implement to do so are presented in 2.3.2.0.1.

```

  Wire.begin(); //Initiating I2C Communication

  setTime(h,min,s,d,m,y); //Setting the exact tame and date with the following
                          //parameters:h,min,sec,day,month,year

  ucg.begin(0); // Initiating UCG lib for the display setting the FONT MODE as SOLID
                //by setting this mode we avoid the blinking when some variable changes it value
  ucg.clearScreen(); // Erasing the last image displayed on the LCD before displaying the new one
  ocsDesign.drawBackground(); //Printing the Solid backGround
  Display_LCD_HomeScreen(); //Displaying the main menu on the display

  pinMode(BUTTON_1, INPUT);  //Configuring pins as inputs for the correct interrupt mode
  pinMode(BUTTON_2, INPUT);
  pinMode(BUTTON_3, INPUT);

  attachInterrupt(digitalPinToInterrupt(BUTTON_1), INTERRUPT_ISR ,CHANGE);   //Defining the interrupts on the
    button, the state funtion
  attachInterrupt(digitalPinToInterrupt(BUTTON_2), INTERRUPT_ISR ,CHANGE);   //where we update the value of the
    button state
  attachInterrupt(digitalPinToInterrupt(BUTTON_3), INTERRUPT_ISR ,CHANGE);    //Triggering the interrupt
    whenever there is a change on the button

  SD.begin(cs_pin); //Initiating SD module
  rf69.init(); //Initiating Radio module
  rf6.setFrequency(FREQUENCY_RADIO_MODULE); //Setting the 433 MHz frequency for the radio module
  rf69.setTxPower(POWER); //Setting the receiver power given that the maximum is 20 dB
```

Code 2.11 – *C++ command for initializing the different modules*

Analogue to how we specified the I2C addresses and the CS pins in the transmitter's code, for some modules it need to be specified on the module's initiation and for other modules it need to be specified on the class calling. The GPS address is the same as indicated in Table 2.4. The radio and the microSD CS pin are also the same presented in Table 2.5. For the LCD module we should note that even though in the SCH () it is specified to be connected through the I2C interface, in reality is connected through the SPI interface. Therefore, when calling the class for this module we have specified the

DC pin (6) or as named in the SCH the A0 pin, the CS pin (7) and the reset pin that as it has been left floating, we have set its value to −1. To finish we also have to mention that they did not include on their SCH () the buttons for controlling the LCD. With the multimeter measuring set to measure continuity we have gather the information about the buttons circuit and the pins to which they are connected. As for the circuit is a simple resistor-capacitor-switch circuit. Regarding the pins they are connected to they are all specified in the *defines.h* file with the following code lines:

```
#define BUTTON_1 5
#define BUTTON_2 4
#define BUTTON_3 3
```

Code 2.12 – *C++ commands for defining the buttons pins*

Once we have initiated all the modules, we are ready for implementing their tasks. Given that we want to keep the main function as clearer as possible, we will proceed like on the transmitter to develop the module's tasks on functions. But before entering to present the most important code lines of the different tasks we have to recall that we have implemented this board's firmware as a FSM thus all the functions will be contained into the states machine function.

2.3.2.0.2 Receiver Loop Function

2.3.2.0.2.1 ISR Routine

The interrupt service routine (ISR) will be the function that we will add into the loop function. This function contains the FSM function. The code lines that this function implements are only the ones needed for reading the button's pin value and for calling the FSM function:

```
btt1 = digitalRead(BUTTON_1); //Reading the button's pin value and storing it into a global variable
btt2 = digitalRead(BUTTON_2);
btt3 = digitalRead(BUTTON_3);

state_function(); //Calling the FSM function which contains the rest of the modules functions
```

Code 2.13 – *C++ commands for defining the ISR routine*

2.3.2.0.2.2 FSM Task

This is actually the main function which contains the tasks of the modules. As we already presented in Figure 2.37, when the variable *state* is set to 0, we are on the first state where the GPS is receiving the coordinates and storing them into the corresponding microSD files, the radio is receiving the packages that the transmitter is sending and the variables that the LCD is displaying are being updated. However, when the *state* variable is set to 1, we only display the corresponding LCD menu depending on the button that has been pressed. For changing the value of the state variable we only need to press any button.

The code lines we have to implement for achieving this behaviour are presented in 2.3.2.0.2.2.

```
    switch(state)
  {
    case 0:
      if (btt3 == HIGH || btt2 == HIGH || btt1 == HIGH ) //Changing the value of the state variable in case any button is pressed
      {
          state = 1;
      }
      else
      {
          state = 0;
      }
      GPS_Receiving_Coordinates_TinyLib(); //Getting GPS coordinates
      SD_write_GPS1_file(); //Storing the coordinates into the .txt file
      rcv_package(); //Receiving the string via radio
      Update_LCD();  //Updating the variables displayed in the LCD
    break;

    case 1:
      if (btt1 == LOW ) //Checking if the button pressed is the button 1
      {
          if (screenNum == 1) //If we are on the middle screen we move to the left screen
          {
              screenNum = 0;
              Display_LCD_GPSScreen();

          }
          else if(screenNum == 2) //If we are in the right screen we move to the middle one
          {
              screenNum = 1;
              Display_LCD_HomeScreen();
          }
      }

      if (btt2 == LOW) //Checking if the button pressed is the button 2
      {
          if (screenNum != 1) //If we are in the left or the right screen we move to the middle one
          {
              screenNum = 1;
              Display_LCD_HomeScreen();
          }
      }
      if (btt3 == LOW) //Checking if the button pressed is the button 3
      {
          if (screenNum == 1) //If we are on the middle screen we move to the right screen
          {
              screenNum = 2;
              Display_LCD_CANSATScreen();
          }
          else if (screenNum == 0) //If we are in the right screen we move to the middle one
          {
              screenNum = 1;
              Display_LCD_HomeScreen();
          }
      }
      state = 0;
    break;
```

Code 2.14 – *C++ commands for implementing the FSM*

As for the function's that we have called on the state 0 for the GPS, the radio module or the microSD are the ones we used on the transmitter so we will refer to the 2.3.1.0.2.3, 2.3.1.0.2.4 and subparagraph 2.3.1.0.2.6 for more information.

Regarding the functions we have used for displaying the image on the LCD module, the main commands needed for programming it are the ones listed on 2.3.2.0.2.2.

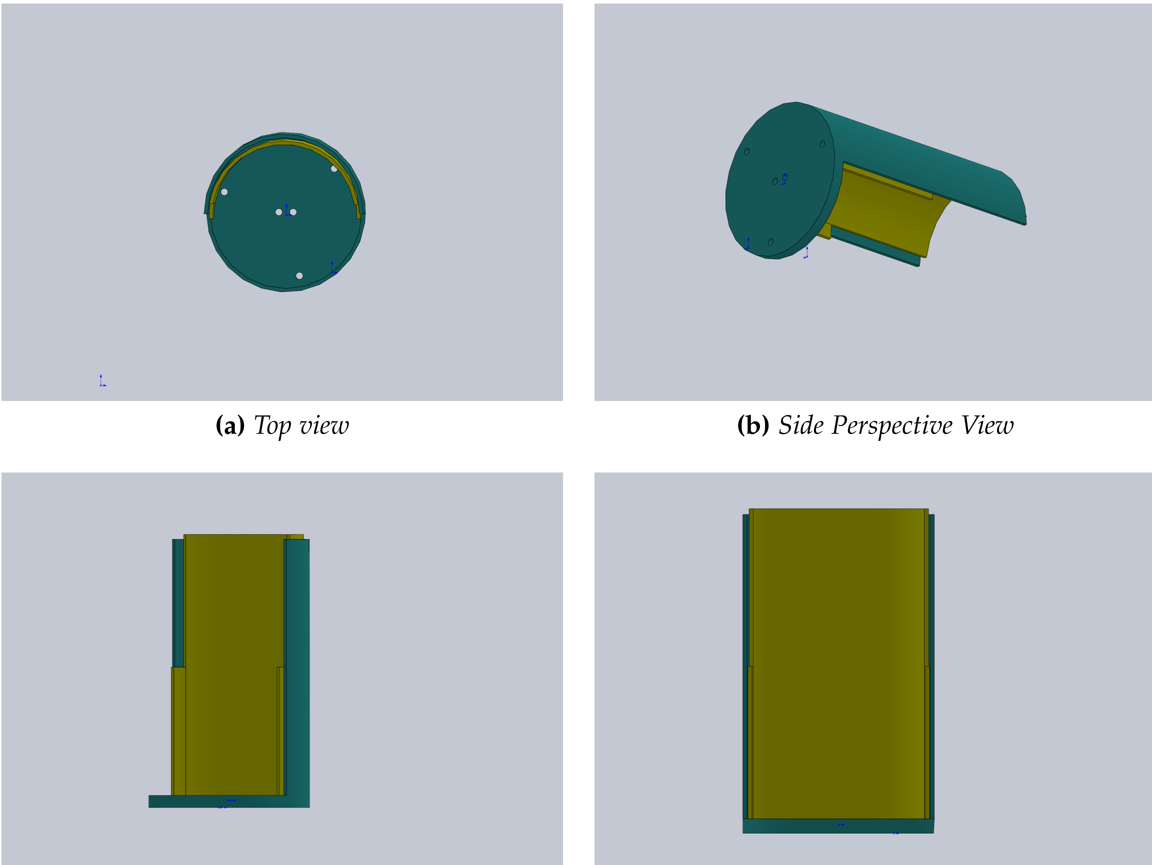
```
ucg.setPrintPos(x,y); //Setting the place on the LCD screen where we want to print something
ucg.setColor(r,g,b); //Setting the color of the thing we are going to be printing
ucg.setFont(ucg_font); //Setting the text font and size in case we want to print some text

ucg.print(); //Printing the thing we want to print (text,variables)

ucg.setRotate90(); //in case we want to rotate the screen 90 degrees
ucg.drawRframe(x,y,w,h,r); //Drawing a round frame
ucg.drawRBox(x,y,w,h,r); //Drawing a round box
```

Code 2.15 – *C++ commands for displaying the menus on the LCD*

We will refer to the Subsection 2.6.2 to see the modules proofs.

2.4 CanSat Assembly

Recalling the main CanSat objective from the Section 1.2, the real challenge is to fit the transmitter board into the volume and shape of a soda can. For this purpose, the can model that the czech engineers have suggested is the one exhibit into Figure 2.38.



(a) *Top view*

(b) *Side Perspective View*

(c) *Front Perspective View*

(d) *Front View*

Figure 2.38 – *OpenCanSat can suggestion [3]*

2.4.1 3D Model

This is not the most optimal design considering that the GPS antenna would be inside the can an the plastic will delay the synchronization. Moreover, as there is no gaps on the can's walls, the LEDs would not be seeing once the board is inside the can. To conclude we should mention that the fitting they have chosen is not the most optimal either since it is not very efficient in order to close the can. It would be necessary to add a rubber that will hold the two can parts and would complication its manipulation.

2.5 Mass Budget

If we remember from Subsection 1.2.1, in this competition there are some requirements that we must met. From those requirements we will highlight the mass limitation. In the CanSat rules [5] it is established that the transmitter must weight between 300 and 350 g including the parachute. Therefore we must be aware of the elements masses through a **mass budget**.

The first thing we need to gather is the modules' masses from their datasheet. The Figure 2.39 presents that information.

Component	Module	Number/PCB	Mass (mg)
uProcessor	SAMD21G18A-AUT	1	300,00
microSD	SanDisk standard uSD	1	672,59
GPS	uBlox-MAX8C	1	600,00
	Ceramic Antenna	1	3400,00
	Exterior Antenna	1	79095,00
Radio Module	RFM69W	1	17970,00
Radio Receiver Antenna	RFM69W	1	15000,00
Pressure Humidity Sensor Temperature	BME280	1	65,00
Pressure and Temp. Sensor	BMP280	1	48,00
Display	LCD 128x160	1	18100,00
Bluetoth	MOD2	1	17900,00
Red LED	0805 SMD	2	9,07
Blue LED	806 SMD	2	4,54
GreenLED	807 SMD	1	4,54
osram LED	2 in 1 Leds	1	6,34
Yellow LED	809 SMD	1	4,54
DC Current Measurement	INA219A	1	76,00
Buck Converter	MCP1826	1	250,00
Boost Converter	MCP73831	1	8,00
Battery	Li-Po 550 mAh	1	10000,00
Pin Header	2x7 pin connector	1	1274
Antenna PCB connector	2 sma right angle connector	2	11334

Figure 2.39 – *OpenCanSat modules' masses*

Then we must compile the case and PCBs weight. Those masses are exhibit in Figure 2.40.

	Number	Mass (g)
CASE (PLA)	1	58,43
ROUNDED PCB	3	24
RECTANGLE PCB	1	18

Figure 2.40 – *OpenCanSat elements' masses*

Finally we can estimate the final product's bulk. For both boards, the final weight is the one showed in Figure 2.41.

	TOTAL (mg)	TOTAL (g)
CanSat Transmitter	37275,04	37,28
CanSat Receiver	172622,07	172,62

Figure 2.41 – *OpenCanSat mass*

Since the receiver board does not have a mass requirement, we have only extracted that information for informational purposes only. However, the mass limitation for the transmitter is not only for not exceeding this bulk but for getting as close as possible to such weight since the rocket trajectory has been calculated by estimating that all CanSats will weight that. As we have estimated a weight of 37.38 g, we will need to add at least 262.62 g of ballast to get near the mass requirements.

2.6 Validation

In this last section we will be showing different pictures from the serial monitor that proofs that the firmware we developed in Section 2.3 works as expected. Also we will be showing pictures and videos of the LCD.

2.6.1 Transmitter Validation

When we upload the whole CanSat transmitter program into the board, we observe that on the serial monitor are printed the sentences showed in Figure 2.47.

```
CANSAT kit INA219 test starting to measure the current with no consumption.

Current time:  15:31:30 4/8/2021
Shunt Voltage: 5.64 mV
Bus Voltage: 4.10 V
Current: 57.10 mA
Load Voltage: 4.11 V
Power = 0.23W

Current time:  15:31:31 4/8/2021

Temperature = 40 °C
Pressure = 917 hPa
Approx altitude = 827 m
Humidity = 19 %

Writing to BME280.txt

Writing to file BME280.txt finished.

Writing to GPS1.txt --> File with the analyzed data from the GPS

Writing to file GPS1.txt finished.

CANSAT kit INA219 test starting to measure the current with the SD and BME280 consumption.

Shunt Voltage: 3.93 mV
Bus Voltage: 4.12 V
Current: 39.00 mA
Load Voltage: 4.12 V
Power = 0.16W

Writing to INA219.txt

Writing to file INA219.txt finished.

Sending: 53 -> 40,917,827,19,3.93,4.12,39.00,4.12,0.16,37.07,-3.60,0.37,7
CheckSum package Sent:72

Packet Received Succesfully
RSSI: -88

Writing to RF69.txt file the string sent and the checksum of the package sent

Writing to file RF69.txt finished.
```

Figure 2.42 – *Open CanSat transmitter's serial monitor*

From the microSD files, we can extract the data presented in Figure 2.43

```
Shunt resistor measured electrical parametres


Current time:  15:21:30 4/8/2021
INA219 measures without any consumption
Shunt Voltage: 4.68 mV
Bus Voltage: 4.22 V
Current: 45.80 mA
Load Voltage: 4.22 V
Power = 0.19W

Current time:  15:21:30 4/8/2021
 INA219 measures with the SD and BME280 consumption
Shunt Voltage: 4.84 mV
Bus Voltage: 4.22 V
Current: 46.30 mA
Load Voltage: 4.22 V
Power = 0.20W

Current time:  15:21:32 4/8/2021
INA219 measures with the SD,BME280,GPS and Radio consumption
Shunt Voltage: 4.08 mV
Bus Voltage: 4.22 V
Current: 41.20 mA
Load Voltage: 4.22 V
Power = 0.17W


Current time:  15:21:32 4/8/2021
INA219 measures without any consumption
Shunt Voltage: 5.73 mV
Bus Voltage: 4.22 V
Current: 57.70 mA
Load Voltage: 4.22 V
Power = 0.24W
```

Figure 2.43 – *Open CanSat transmitter's INA219.txt file*

```
CANSAT measured data

Current time:  15:21:30 4/8/2021
Pressure = 917 hPa
Temperature=39 °C
Approx altitude = 826 m
Humidity = 19 %

Current time:  15:21:32 4/8/2021
Pressure = 917 hPa
Temperature=39 °C
Approx altitude = 826 m
Humidity = 20 %

Current time:  15:21:34 4/8/2021
Pressure = 917 hPa
Temperature=39 °C
Approx altitude = 826 m
Humidity = 19 %
```

Figure 2.44 – *Open CanSat transmitter's BME280.txt file*

```
Date / Time:  8/5/2021    15:21:48.0
Latitude ---> 37.07,     Longitude ---> -3.60
Speed: 0.61 kn.

Altitude: 881.80 m.
Satellites reached: 7.

Date / Time:  8/5/2021    15:21:48.0
Latitude ---> 37.07,     Longitude ---> -3.60
Speed: 0.61 kn.

Altitude: 881.80 m.
Satellites reached: 7.

Date / Time:  8/5/2021    15:21:55.0
Latitude ---> 37.07,     Longitude ---> -3.60
Speed: 0.18 kn.

Altitude: 881.80 m.
Satellites reached: 7.

Date / Time:  8/5/2021    15:21:55.0
Latitude ---> 37.07,     Longitude ---> -3.60
Speed: 0.18 kn.

Altitude: 881.80 m.
Satellites reached: 7.
```

Figure 2.45 – *Open CanSat transmitter's GPS1.txt file*

```
$GNRMC,132148.00,A,3704.48645,N,00336.06430,W,0.615,,050821,,,A*
71
$GNVTG,,T,,M,0.615,N,1.139,K,A*35
$GNGGA,132148.00,3704.486
45,N,00336.06430,W,1,07,3.10,881.8,M,46.7,M,,*58
$GNGSA,A,3,03,
```

Figure 2.46 – *Open CanSat transmitter's GPS2.txt file*

Note that the string received from the GPS which is printed in Figure 2.46 give us the information written in Figure 2.45. The GPS strings are analyzed in Section E.1 therefore we will refer to that section for further information.

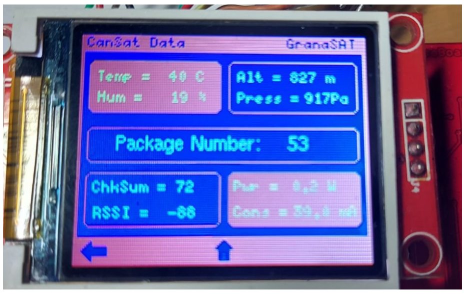
2.6.2 Receiver Validation

As for the receiver we need to mention that the radio antenna and the GPS antennas were not included on the kit, therefore we used some antennas available on the laboratory. To check whether this antennas were the right fit for their device we follow the procedure described in Section C.2 for measuring the antennas RF parameters with a VNA.

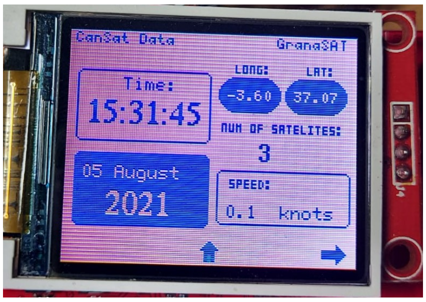
```
Writing to file GPS1.txt finished.

Package Received: 53 -> 40,917,827,19,3.93,4.12,39.00,4.12,0.16,37.07,-3.60,0.37,7
CheckSum package Sent:72
RSSI: -88
Sent a reply

Writing to RF69.txt file the string sent and the checksum of the package sent

Writing to file RF69.txt finished.
```

Figure 2.47 – *Open CanSat transmitter's serial monitor*

(a) *LCD main menu*



(b) *LCD transmitter menu*



(c) *LCD receiver's menu (GPS)*

Figure 2.48 – *LCD captures after uploading the receiver software*

Chapter 3

System Analysis

This third chapter tackles another phase related to analysis and requirements definition. It resembles the more general 'System Analysis' and will cover the whole process between the requirements redefinition to the modules selection.

We already studied in Subsection 1.2.1 the basics requirements that our device must have to enter the competition, but since the Open CanSat kit deigned by the Czech engineers implements some features extra, we need to implement those features as well to be able to compete against their product in the market.

During this chapter we will pursue a deep study of the different technologies and modules that will cover the objectives and requisites established, not only on the preliminary requirements defined in Chapter 1, but also the new requisites we will be defining after the prior art study in Chapter 2 according to the **reverse engineering process** described in Section 1.3. By the end of this chapter, the project may go through the **System Design** process, detailed in Chapter 4.

3.1 Requirements Engineering

After studying the requisites a CanSat must met in order to enter the competition from the Esero document rules [5], we can conclude that our CanSat must accomplish the following conditions. Note that this requirements were briefly specified in Subsection 1.2.1.

- Total weight: 280 g [1]
- Length: 1.5 m

[1] This values (weigh, length,diameter) are dependent on the rocket, do this value may change from one year to another

- Diameter: 79.4 m
- Measuring: Air temperature and air pressure
- Sending the parameters measured via radio once per second
- Developing a Secondary mission that demonstrate an innovative mission with some scientific or technological value

However, during the prior art analysis conducted in Chapter 2 we need to add to that list the features listed below to be able to compete against their product in the market.

- Measuring the altitude and air humidity
- Tracking the GPS position to identify where the CanSat lands
- Saving the measured data on a microSD file
- Measuring the device's current and voltage
- Protecting the device from the possible surges derived from the human's manipulation
- Being able to send the data received from the radio by bluetooth
- Displaying the data in a LCD

3

Being established the competition's main requirements and the extra features that the Czech engineers have included, we need to find some modules capable to implement all of this features which improve not only the product's efficiency but also the price. Therefore we need to make a market study to see which modules allow us to achieve this two objectives. To do so, what we are going to be doing on the following sections is to show a trade-off figures that allow us to compare the modules they have chosen to the one available on the market.

3.2 Electronic Architecture

The main module that we need to think of is the microprocessor. Once we have selected the best microprocessor for our device, we can proceed to choose the rest of the modules depending on the pins and interfaces our microprocessor implements.

For choosing the best modules we will be showing some of the ones we think are more interesting and trading the off. The modules we have chosen are not only the ones we have found online but also some of the modules our providers have offered us. That implies that we have not only made a online market study but also have talked with some providers to learn more closely the selling prices of the small businesses.

3.2.1 uProcessor Selection

In paragraph 2.1.1.1.1 we learned that the microprocessor they have used is a single core. That implies that we only can implement a super loop architecture for programming, entailing the related delay. As one of the requirements is to send the data every second, that something that we need to take into account.

Another thing we studied is the little memory space this microprocessor implements. That could limit not only the modules we can incorporate but also imply some delays.

The last important thing we have to take into account are not only the interfaces that the microprocessor can develop but also the GPIO pins that include.

Taking into account that we have a space limitation as it has to fit into a soda can, we should be thinking of using a microprocessor that incorporates a bluetooth feature at least. Moreover we should be looking microprocessor that can implement a RTOS architecture since we also have a time limitation.

For that reason and keeping in mind that we have to be close to their price, we will be considering using a microprocessor such as the **ESP8266 12E** or the **ESP32-WROOM-32D**. For choosing which one is more suitable for our application, we will make a table trade-off:

	SAMD21G18A-AUT	ESP-32	ESP-12
CPU	ARM CORTEX -M0+CPU running at up to 48 MHz: - Single-cycle hardware multiplier - Micro Trace Buffer (MTB)	Xtensa LX6 32 bits Single/Dual core Up to 600 MIPS	Tensilica L X106 32 bits Single core
SRAM	32 kB	520 kB (16 kB SRAM in RTC)	< 50 kB
FLASH	256	4 MB (larger available)	4 MB
ROM	No	448 kB	No Porgrammable ROM
POWER SUPPLY	1.62 to 3.63 V	2.3 to 3.6 V	3.0 to 3.6 v
CURRENT CONSUMPTION	85 to 1.2 mA	Max: 225 mA Typ:80mA Sleep: 2.5 µA (10 µA RTC + memoria RTC) Low Consumption: >150 µA	Max: 225 mA Typ:80mA Sleep: 20 µA (RTC + memoria RTC)
TEMPERATURE RANGE	-40 to 125 ºC	-40 to 125 ºC	-40 to 125 ºC
Wi-Fi	No	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 150 Mbps	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 72.2Mbps
Wi-Fi MODES	-	Station / SOFTAP / SOFTAP + STATION / P2P	Station / SOFTAP / SOFTAP + STATION / P2P
NETWROK PROTOCOL	-	Ipv4 / Ipv6 / SSL / TCP / UDP / HTTP / FTP / MQTT	Ipv4 / TCP / UDP / HTTP / MQTT
BLUETOOTH	No	V4.2 BR/EDR+BLE	No
ETHERNET MAC INTERFACE	-	10/100 Mbps	No
HW ENCRYPTATION	No	Yes	No (TLS 1.2 in SW)
GPIO	38	36	17
DAC	10 bit	2 DAC channels 8 bit	NO
ADC	12 bit Up to 20 Channels	SAR 12 bits 8 Channels	SAR 10 bits 1 Channel
INTERFACES	DMAC / TC / TCC / RTC / WDT / CRC / SERCOM / USART / 12C / SPI / ADC / DAC / AC / PTC	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / DAC / TOUCH / PWM / LED	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / PWM / LED
TOUCH SENSOR	Yes	Yes (8 Channels)	No
TEMPERATURE SENSOR	No	Yes	No
HALL EFFECT SENSOR	No	Yes	No
DIMENSIONS	7 x 7 x 1.0 mm^3	57 x 28 x 1 mm^3 8 g	16 x 24 x 3.4 mm^3 1.72 g
PRICE	2.73 €	2.20 € https://www.mouser.es/	3 – 6€
PREFERENCE ORDER	3	1	2
ELECTION	✗	✓	✗

Figure 3.1 – *Microprocessor trade-off*

Note that for both microprocessor, the one that the Czech egineers have chosen and the one we have selected, it is possible to restore the damage bootloader using a J-Link even though the **ESP32-WROOM-32D** is more protected and less likely to be damaged than the **SAMD21G18A-AUT**. Restoring the bootloader in the ESP32-WROOM-32D is described in [67] while restoring the bootloader is explained in Section D.1.

Another important feature we must consider is the USB to serial conversion in order to being able to talk to the microprocessor. On the SAMD21G18A-AUT microprocessor, we studied in paragraph 2.1.1.1.9 that it could be directly interfaced since this microprocessor integrates this conversion. However, neither the ESP32-WROOM-32D nor the ESP8266 12E microprocessors integrates this conversion. Therefore we need a chip that will allow us to talk to the microprocessor from the PC. For that matter we have chosen the **CH340C** chip since is the one available on the laboratory. This module is a really cheap module (could be found from 0.44 €) that could allow us to do this conversion with a small consumption and high velocity.

The last important feature we must mention is that as the ESP32-WROOM-32D microprocessor implements bluetooth we would be including one of the requriments the Czech engineers included without having to add any more modules. Moreover, the WiFi feature would give us some extra possibilities to implement for the secondary mission.

3.2.2 Temperature, Humidity and Pressure Sensor Selection

The main module for entering the competition is a temperature and pressure sensor given that one of the requirements is to measure those parameters. Moreover, the Czech engineers have introduced measuring two additional parameters such as the humidity and the altitude. Hence we need to find the best solution for integrating a sensor capable of measuring those parameters with a cheap a small module.

Studying the best modules suitable for our device on the online business as well as in our providers list, we have come up with the table trade-off showed in Figure 3.2.

By observing the Figure 3.2, we can conclude that the best sensor for our application is **BME280** in light of the fact that we could be measuring the three parameters with a small and economical sensor. Another important feature that this sensor provide us is not only its high consumption but also high precision.

	DHT11	BMP280	BME280	DS18B10
SUPPLY VOLTAGE	3 to 5.5 V	1.2 to 3.6 V	1.2 to 3.6 V	3.0 to 5.5 V
CURRENT	Active Current: 1 mA Standby Current: 150 µA	Active Current: 4.2 µA Standby Current: 0.1 µA	Active Current: 1.4 mA Standby Current: 0.5 µA	Active Current: 0.6 mA Standby Current: 0.75 µA
TEMPERATURE RANGE	0 to 50 ºC	-40 to 85 ºC	-40 to 85 ºC	-55 to 125 ºC
PARAMETRES MEASURE	Temperature Humidity	Temperature Pressure	Temperature Pressure Humidity	Temperature
ACCURACY	Temperature: ±2 ºc Humidity: ±5 %	Temperature: ±1 ºc Pressure: ±1 hPa	Temperature: ±1 ºc Pressure: ±1 hPa Humidity: ±3 %	Temperature: ±0.5 ºc
RESOLUTION	Temperature: 1 ºc Humidity: ±1 %	Temperature: 0.01 ºc Pressure: 0.16 Pa	Temperature: 0.01 ºc Pressure: 0.18 hPa Humidity: 0.008 %	Temperature: ±0.5 ºc
DIMENSION	12 x 15.5 (+ 8 mm pin) mm^2	2.5 x 1.75 x 0.95 mm^3	2.5 x 2.30 x 0.93 mm^3	3.94 x 4.95 x 4.95 mm^3
PRICE	4.24 € mouser.es	3.175 € es.rs-online.com	4.90 € mouser.es	7.97 € es.farnell.com
PREFERENCE ORDER	3	2	1	4
ELECTION	✗	✗	✓	✗

Figure 3.2 – *Temperature measuring modules trade-off*

It should be noted that in *aliexpress* we have found a module that implements this sensor and the extra elements (resistances and capacitors) that this module requires for working with a better price than buying the sensor separately. However, the PCB this module has is too big for our device.

3.2.3 MicroSD Card Socket Selection

Given that this is a pretty simple module, we will be choosing a module that can provide us two main attributes:

- Make soldering easier
- Detecting whether there is a card inserted or not

In fact, the microSD socket they have used give us those features even though they have not used the detecting card mechanism. However, the socket they have used is a bit overprice since we can find modules that achieve both our goals for half the price.

Focusing on that goal, reducing the price and maintaining the same features, we have come up with the next trade-off table showed in Figure 3.3 for choosing the best module. In our case, we will only be comparing the one they have used (**MOLEX 502774**) with the one we think is best (**MOLEX 504528**).

	MOLEX 502774	MOLEX 504528
MAXIMUM CURRENT	0.5 A	0.5 A
TEMPERATURE RANGE	-25 to 85 ºC	-30 to 85 ºC
TYPE	Push-Push	Pin-Eject
CARD DETECTION	Yes, Open Circuit	Yes, Open Circuit
DIMENSION	16 x 14.3 x 1.8 mm^3	16 x 18.25 x 1.4 mm^3
PRICE	3.20 € es.rs-online.com	2.77 € mouser.es
PREFERENCE ORDER	2	1
ELECTION	✗	✓

Figure 3.3 – *microSD sockets trade-off*

From this trade-off table we can conclude that the best option for our device is **MOLEX 504528** as it is cheaper than the one they used.

3.2.4 Radio Module Selection

One of the most important modules is the radio module as it is needed for achieving the primary mission requirements. This module must assure a communication distance of at least, 1000 m since that is the distance the rocket is going to reach before releasing the CanSats.

We noticed from the study we did on the last chapter, that the distance of the module they chose is not the most convenient as it only allow us to maintain the communication at a maximum distance of 500 m. Therefore we can not keep this module and should be replacing it.

Taking into account this feature, we will be implementing another table for trading-off in Figure 3.4. The most important thing we will be focusing in this case is the distance of communication but we will not be forgetting about the sensitivity, the modulation or the link budget.

From the Figure 3.4 we can conclude that the module we will be choosing is the **RFM69W**. With this new module we will be achieving the objective of maintaining the communication with a 1000 m distance but also we will be reducing the radio module consumption as well as reducing the space for this module.

	RFM69W	SX1278	RFM96W	HM-TRP
POWER SUPPLY	2.4 to 3.6 Vdc	1.8 to 3.7 V	1.8 to 3.7 V	2.4 to 3.6 V
CURRENT SUPPLY	Receive mode : 16 mA Trsmit mode : 16 – 45 mA Sleep mode : < 1 uA Santdby mode : <1.5 mA	Receive mode : <12 mA Trsmit mode : < 20 mA Sleep mode : < 1 uA Santdby mode : <1.8 mA	Receive mode : < 12.1 mA Trsmit mode : < 120 mA Sleep mode : < 1 uA Santdby mode : < 1.5 uA	Receive mode : < 30 mA Trsmit mode : < 120 mA Sleep mode : 2 uA
FREQUENCY FUCTIONING	433 MHz	433 MHz	433 MHz	414 to 454 MHz ISM band
DISTANCE	500 m	5000 m	2000 m to 20 km depending on the antenna	Over 1 km in open air
TEMPERATURE RANGE	- 20 to 70 ºC	-40 to 85 ºC	-40 to 85 ºC	-40 to 85 ºC
SENSITIVITY	-120 dBm	Down to -148 dBm	-126 dBm	-117 dBm
MODULATION	FSK , GFSK , MSK , GMSK or OOK	FSK, GFSK, MSK, GMSK, OOK	FSK, GFSK , MSK , GMSK , OOK	FSK 2 way half-duplex communication, strong anti-interfere
OTA DATA RAGE	1.2 to 300 kbps	1 to 300 kbps	100 bps to 1 Mbps	1.2 kbps to 115.2 kbps
LINK BUDGET	115 dB	168 dB	-148 dBm	
INTERFACE	Serial SPI	Serial SPI	Serial SPI	Standard TTL UART, extendable to RS232 or other interface
TEMPERATURE SENSOR	Yes	Yes	Yes	No
LOW BATTERY DETECTOR	No	Yes	Yes	Yes
DIMENSIONS	19.7 x 16 x 1.9 mm^3	17.2 X 17 mm^2	16 x 16 x 2 mm^3	16 x 20 x 2 mm^3
PRICE	1.52 € https://es.aliexpress.com/	3.13 € https://es.aliexpress.com/	3.26 € https://es.aliexpress.com/	7.25 € www.mecter.com
PREFERENCE ORDER	4	3	1	2
ELECTION	✗	✗	✓	✗

Figure 3.4 – *Radio modules trade-off*

3.2.5 GPS Module Selection

Even though the GPS module is not a primary mission requirement, is a great idea to implement in case the landing is not where expected. With this module we could find the CanSat more easily if we send its coordinates via radio.

The reason for including this module is not only for finding the CanSat more easily but also as the Czech engineers have include it on their kit. Even though the one they have chosen is a good module, we find it slightly expensive for the use we are going to give it.

In fact, in this case we are also going to make a table trade-off but we will be focusing on the price and size of the device as all this kind of modules implement similar accuracy and sensitivity characteristics on this price range. This table is the one exhibit in Figure 3.5.

Focusing on the price and size of the device, we will be choosing the **ATGM336H** GPS module as shown in Figure 3.5. With this module we will be worsening the device size but we will be improving the price.

This module requires an antenna so that it can communicate with the satellites. They have chosen a ceramic antenna which we think is the most practical module for our device. Since the one the have used is discontinued, we will be searching a new economical ceramic antenna. We will not be making a trade-off table for this module since they have the same characteristics and the only think it can change is the size and price.

	Ublox-MAX8C	**ATGM336H**	**CAM-M8C**	**BG01-H111S100**
POWER SUPPLY	1.65 – 3.6 V	2.7 to 3.6 V	1.65 TO 3.6 V	Typ. 3.3 V
POWER CONSUMPTION	Low power consumption	StandBy : <25 mA	71 mA	Acquisition Current: 30 mA Tracking Current: 20 mA
TEMPERATURE RANGE	-40 °C to 85 °C	-40 °C to 85 °C	-40 °C to 85 °C	-40 °C to 85 °C
UPDATE RATE	0.25 Hz to 10 MHz	1 Hz to 10 Hz	10 Hz	
ACCURACY	Horizontal position <4 m Velocity : <0.05 m/s Time: 60 ns	Horizontal position: <2 m Velocity : <0.1 m/s Time : < 30 ns	Horizontal position: Velocity : <0.05 m/s Acceleration:	Horizontal position: 2.5 m Altitude position: 3.5 m Velocity : 0.1 m/s Time pulse signal: 30 ns
INTERFACES	UART , DDC (I2C fast mode compatible)	UART1/UART2/I2C	UART / DDC / I2C	UART
SENSITIVITY	Navigation : -166 dBm Tracking : -166 dBm Reacquisition : -160 dBm	Navigation : -162 dBm Tracking : -162 dBm Reacquisition : -162 dBm	Navigation : -164 dBm Tracking : -164 dBm Reacquisition : -160 dBm	Cold Start : -148 dBm Warm Start: -162 dBm Tracking : -166 dBm Reacquisition : -164 dBm
DYNAMIC PERFORMANCE	Maximum Altitude: 50000 m Maximum Valocity : 500 m/s Maximum Aceleration: < 4 G	Maximum Altitude: 18000 m Maximum Valocity : 515 m/s Maximum Aceleration: < 4 G	Maximum Altitude: 50000 m Maximum Valocity : 500 m/s Maximum Aceleration: < 4 G	Cold Start: 27.5 s Warm Start: < 1 s Re-Acquisition: < 1 s A-GNSS: < 1 s
DIMENSIONS	9.7 x 10.1 mm^2	10.1x 9.7 x 2.4 mm^3 0.6 g	9.60 x 14 x 2 mm^3	16.2 x 12.2 mm^2
PRICE	5.62 € https://es.aliexpress.com/	2.77 € https://es.aliexpress.com/	Not Available	6.75 € www.mecter.com
PREFERENCE ORDER	2	1	4	3
ELECTION	✗	✓	✗	✗

Figure 3.5 – *GPS modules trade-off*

3.2.6 Display Module Selection

Remembering the requirements that the Czech engineers have introduced we have to think of adding a display into our base board design for displaying the data. As we already mentioned in paragraph 2.3.2.0.1, even though they have not included them on their schematic, they have added three buttons for controlling their display.

On this board we do not have a size limitation, therefore we can think of adding a bigger display since the one they have used is small according to our opinion. Additionally, the price of their display is higher than we expected. Consequently, in Figure 3.6, which is the table for trading-off some modules, we will be focusing not only on the price but also on the size.

	LCD + SD 8TFT 128x160	SSD1331 OLED	SSD 1306 OLED	ILI9341	TFT LCD MODULE-HT0280CI01BR1
SUPPLY VOLTAGE	-0.3 to 4.6 V	2.4 to 3.5 V	1.65 to 3.3 V	2.5 to 3.3 V	2.5 to 3.3 V
CURRENT		Sleep Mode: 10 uA Max Supply Current: 500 uA	Sleep Mode: 10 uA Max Supply Current : 150 uA	80 mA	80 mA
TEMPERATURE RANGE	-30 to 85 ºC	-40 to 85 ºC	-40 to 85 ºC	-20 to 70 ºC	-20 to 70 ºC
RESOLUTION	128 x 160 1.8”	96 x 64 0.95”	128 x 64 0.96”	240 x 160 2.8”	240 x 320 2.8”
COLORS	65 thousand RGB format	65 thousand	Monochrome	RGB Format	RGB Format
TOUCH PANNEL	No	No	No	Yes	Yes
DIMENSION	58 x 34 mm^2	30.7 x 27.3 x 11.3 mm^3	26.7 x 19.26 x 1.85 mm^3	56 x 35 x 1.41 mm^3	50 x 69.2 x 3.45 mm^3
PRICE	21 $ www.hotmcu.com	5.20 € https://es.aliexpress.com/	7.99 € https://es.aliexpress.com/	12.76 € https://aliexpress.com	7.25 € www.mecter.com
PREFERENCE ORDER	3	4	5	1	2
ELECTION	✗	✗	✗	✓	✗

Figure 3.6 – *Display modules trade-off*

In this instance, we will chose the **ILI9341** device since it reduces greatly the price and improves the size. Another important thing we will be highlighting from our election is that this device implements a touch panel thus we won't need to add any extra button that increases the price and decreases the space.

3.2.7 Current \Power Monitor Selection

Keeping in mind that the product we are using is for electronics beginners in most cases, they are not too aware of what is the consumption or why it should be taken into account when designing any device. This module can initiate them into researching about this characteristic. For that reason, we will be looking for a economical module that does not take much space. We do not want a high precision monitor but a module that can give us the right precision.

After a research, we have noticed that some of the modules requires an extra ADC module for it to work that would require more space and a price increase.

When comparing some modules we have come up with the trade-off table exhibit in Figure 3.7.

	INA168	INA219	INA231
SUPPLY VOLTAGE	2.7 to 30 V	0.3 to 6 V	2.7 to 5.5 V
INPUT CURRENT	10 mA	5 mA	5 mA
TEMPERATURE RANGE	-40 to 125 ºC	-40 to 125 ºC	-65 to 150 ºC
SHUNT VOLTAGE	100 to 500 mV	0 to ±320 mV	-81.92 to 81.9175 mV
ADC BASIC RESOULUTION	-	12 bits	16 bits
INTERFACE	-	I2C	I2C
INTEGRATED ADC	No	Yes	Yes
DIMENSION	3 x 3.05 x 1.45 mm^3	3 x 3 x 1.45 mm^3	1.675 x 1.418 x 0.4 mm^3
PRICE	2.58 € mouser.es	2.22 € mouser.es	2.74 € mouser.es
PREFERENCE ORDER	3	1	2
ELECTION	✗	✓	✗

Figure 3.7 – *Current \Power Monitor modules trade-off*

We may think that the best module is the **INA231** is the best choice in this case since its smaller and does not raise the price too much, however this module is way too small for manual soldering. Just as observed in Figure 3.7, we can conclude that the best module for our application is the **INA219** due to the I2C interface and the economical price for a small module.

3.2.8 LDO Regulator Selection

If we go back to paragraph 2.1.1.1.9, we can observe they have used two LDO regulators, one for boosting the 3.7 V tension to 5 V and another one for booting it into 3.3 V. Since boosting the tension is a more complicated thing than stepping it down an all our modules can function with 3.3 V, we will be using only one buck converter to turn the 3.7 V battery's tension into the 3.3 V supply tension.

Before choosing the perfect boost converter for our device we need to first develop a power budget in order to be able to supply correctly our modules. This power budget is the one showed in Subsection 4.1.3 and restricts our search by making us look into buck converters that can provide us a current of at least 700 mA. Therefore, we can no longer use the one they have used since it provides only 1 A and on the worst case scenario where all modules work at it maximum consumption, we will be requiring a 1.9 A current.

Looking into our provider list and into the online suppliers, we can come up with the following trade-off table in order to choose the most suitable boost converter for our application.

	MPC1826S	LM3671	AP3429/A
SUPPLY VOLTAGE	2.3 to 6.0 V	0.3 to 6 V	2.7 to 5.5 V
OUTPUT CURRENT	1 A	600 mA	2 A
TEMPERATURE RANGE	-40 to 125 ºC	-40 to 125 ºC	-65 to 85 ºC
CONMUTATION FREQUENCY	-	2.6 MHz	1 MHz
DIMENSION	10.67 x 15.87 x 4.8 mm³	2.00 x 2.0 mm²	2.9 x 1.6 x 1.0mm³
PRICE	- mouser.es	1.36 € mouser.es	0.36 € mouser.es
PREFERENCE ORDER	3	2	1
ELECTION	✗	✗	✓

Figure 3.8 – *Buck Converter modules trade-off*

Finding a buck converter that can provide at least 1.9 A is not an easy task, in fact, we could only find one with a suitable price and size. We have also explored the possibility of using two buck converters to reach the desired supply current. However, examining the Figure 3.8, we conclude that the most economical solution is using only one module by introducing the **AP3429 \A** buck converter in our design.

3.2.9 USB Type Selection

For this module we will be focusing on two main characteristics: the size and speed. There are several of USB connectors, from 1998 to 2019 they have improving the USB speed from 12 Mbps to 40 Gbps. This connection speed could be integrated by the USB type-A, type-B or the type C but the latest generations (3.2) are more likely to use the full advantages in a type-C USB connector.

Taking this into account, we will be referring to Figure 3.9 for selecting the most suitable USB type depending mostly on their size and price. Since we do not need a super fast connection, we will not take that feature so much into account.

3

	TYPE-A	TYPE-B	TYPE-C
STANDARD	USB 1.1,USB 2.0,USB 3.2	USB 1.1,USB 2.0,USB 3.2	USB 1.1,USB 2.0,USB 3.2,USB 4
MAXIMUM DATA TRANSFER SPEED	12 Mbps to 10 Gbps	12 Mbps to 10 Gbps	12 Mbps to 40 Gbps
DIMENSION	13.10 x 17.70 x 11.75 mm^3	Type B: 12.04 x 7.78 x 21.60 mm^3 Micro B: 3 x 3 x 1.45 mm^3	8.94 x 8.37 x 3.14 mm^3
PRICE	2.73 € mouser.es	Type B: 2.20 € Micro B: 0.594 € mouser.es	2.74 € mouser.es
PREFERENCE ORDER	3	1	2
ELECTION	✗	✓	✗

Figure 3.9 – *USB connectors trade-off*

As the serial communication is not an important fact that we should be worrying about, we will choose the **USB micro-B** in light of the fact that it is the most affordable and smallest connector.

3.2.10 Battery Type Selection

The last module we need to think of is the battery. There are two types of rechargeable batteries that can supply 3.7 V and they are distinguished by their form: one is cylindrical and the other one is flat. Since both of them are made from lithium polymer, we will only be focusing on the size and price.

For choosing the best battery we must focus not only on getting 3.7 V tension but also on the capacity of the battery. This is an important feature to take into account since the price will depend on that feature. For choosing the most suitable battery for our device, we will have to analyze the device's total consumption for selecting this parameter according to the time we would want to be ON before the battery discharges.

We will be referring to Subsection 4.1.3 for observing what is the total consumption. Since our device requires at least 700 mA, we will need a 700 mAh battery for making sure that the device is ON during an hour for synchronizing the device and fulfilling the mission before the battery discharges.

	Cylindrical Li-On Battery	**Flat Li-On Battery**
SUPPLY VOLTAGE	3.7 V	3.7 V
CAPACITY	800 mAh	800 mAh
REFERENCE	18650	383450
PROTECTION CIRCUIT	No	Yes
DIMENSION	67 (length) x 18 (diameter) mm^2	50 x 34 x 3.8 mm^3
PRICE	1.81 € es.aliexpress.com	2.66 € es.aliexpress.com
PREFERENCE ORDER	2	1
ELECTION	✗	✓

Figure 3.10 – *Batteries trade-off*

For selecting the most suitable battery for our application we must think of how are we going to introduce the boards on the can. Given that the cylindrical battery length is larger than the can diameter, we could only fit this battery if we structured the battery parallel to the case wall. That would severely affect the modules space. As we think is best to keep the circular PCB boards, we will be choosing the flat battery. Concluding, in our design we will be introducing a **Flat Li-On Polymer Battery** of **700 mA**.

3.2.11 Battery Charger Selection

Once we have selected the battery, we run into the need of thinking how we are going to charge it since this is not a one-use device.

The one the Czech engineers used is great module but it is only capable of supplying 500 mA and, since we already know from Subsection 4.1.3 that we will need more current, we need to find a more suitable module for our device.

The best module we could though of is the **TP4056** for their dimensions and for its price is a USB charger module that allows easy charging single-cell Li-Po batteries that works with a nominal voltage of 3.7 V and full charge at 4.2 V. The charging current is 1 A, but this value can be modified by changing one of their resistor as shown in the datasheet [56].

For more information about this module or ist circuit we will refer to paragraph 4.1.1.3.2.

3.3 Mechanical Architecture

3.3.1 PCBs shape

The last section has been useful for selecting the most suitable modules for our application as well as for giving us an idea of how many sensors we are going to be implementing and the size of them.

On this section we will be looking into the fact of how are we going to be fitting them into the can and how are we going to improve the can design.

There are two possibilities for fitting the modules on the can as we already discuss in Subsection 3.2.10. One possibility is to fit them in a rectangular PCB which will be introduce with the shortest side colliding with the cylindrical base as shown in Figure 3.11. The other possibility is to fit all the modules in a circular PCB as exposed in Figure 3.12 analogue to how the Czech engineers have done it.

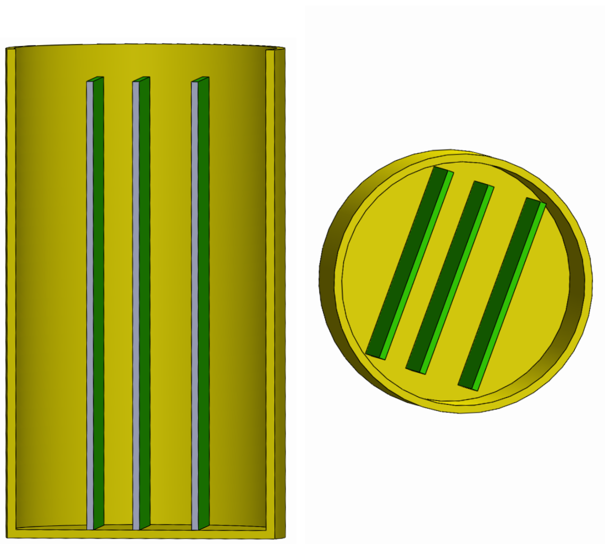


Figure 3.11 – *Rectangular PCB structure*

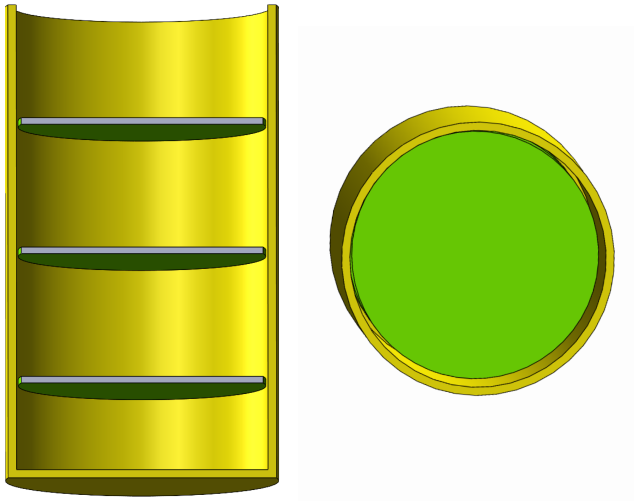


Figure 3.12 – *Circular PCB structure*

From both possibilities we may think that we can introduce a third possibility as an hybrid of both of them but is not the best idea. Implementing a circular PCB with a rectangular one means that for transferring the signals from one PCB to the other we must use some wires that may become loose during the rocket launch.

There are two main characteristics that will make us select the one we are going to be implementing in our design. The first on is the space between boards. For connecting the boards we have to use some pin header which are known for being very voluminous. Using this module would reduce the possibility of having 3 boards to have only 2. Another voluminous module we are thinking of adding is a buzzer. Therefore we will only be having teo boards. Moreover, fitting a rectangular board in a circular space takes up certain space making it less appropriate for our device. Additionally to that fact, we need to think of orienting the GPS antenna vertical to the satellites. This could only be achieve placing this module on the outer circular board, not on the can side.

The other thing we may analyze is the area available for placing the modules. For analyzing this feature we will need to be calculating their area. The two equations we will be using for calculating their area are:

$$CircularBoard_{Area} = \pi \, r^2 \tag{3.3.1}$$

$$RectangularBoard_{Area} = L \, W \tag{3.3.2}$$

For fitting into a can shape, we have to size the boards with: $r = 30$ mm, $L = 51$ mm and $W = 106$ mm. Therefore, the areas are:

$$CircularBoard_{Area} = 2827 \text{ mm}^2$$

$$RectangularBoard_{Area} = 5406 \text{ mm}^2$$

In view of the fact that we could only be using two rectangular boards opposing the three circular boards we could use, the total are available for both possibilities is:

$$CircularBoard_{Area} = 8482,3 \text{ mm}^2$$

$$RectangularBoard_{Area} = 10812 \text{ mm}^2$$

Although with the circular PCB boards we will have less area available, we will choose this design in light of the fact of being able to fit a buzzer in our design.

3.3.2 CanSat Can model

3

This last section of this chapter tackles the problems we may experience with the Czech engineers CanSat can model and how we may improve their design. Although we may think that designing a can is not that complicated, there are several aspects that need to be taking into account for bettering the device efficiency and reducing the possible issues we may experience.

One of the things they did not think about when designing the can model is the fact that for receiving the GPS information the antenna must have a gap to the air. Even though the plastic wall is thin, there is a possibility that it blocks off the correct communication between the GPS module and the satellites.

Another important feature to think off when designing the can is the fact of making some gaps on the can body to be able to see the different LEDs as an indicator that the device is still ON. Moreover, reaching the device switch is also a great idea to be considering.

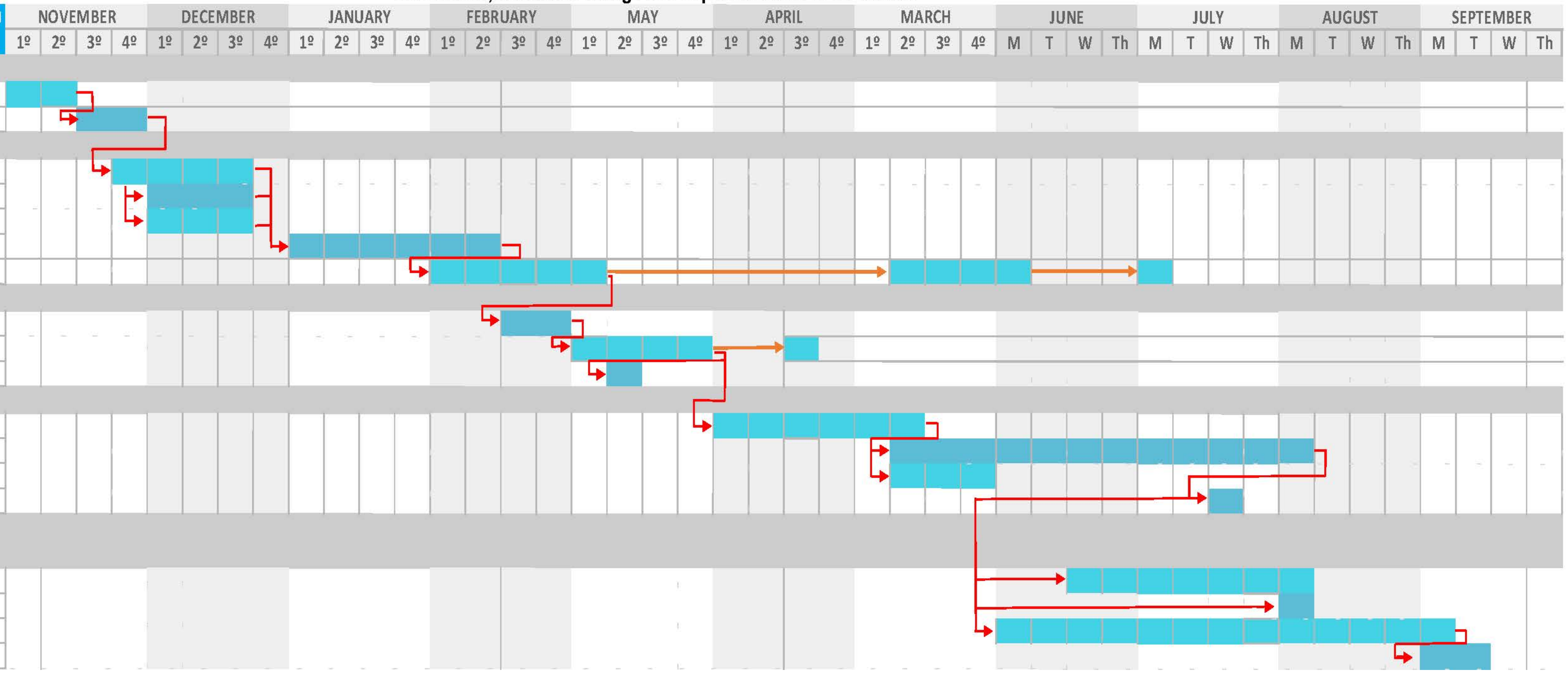
The last important detail that we need to pay attention to is the holes for the parachute. Placing them too closely as they have done would make us loose stability making the CanSat swing.

In the next chapter we will be developing our system design, therefore we would be taking into account all this features to design the most suitable can that will allow us to compite against the Czech engineers on the market.

3.4 Gantt Diagram

Mechanical, Electronic Design and Implementation of a CANSAT

TASK NAME	START DATE	END DATE	DURATION (WORK
Project's Start			
Project's Election	01-nov	11-nov	9
General Objectives	11-nov	30-nov	14
Reverse Engineering			
Information Collection	30-nov	18-dic	14
System Objectives	01-dic	18-dic	13
Functional Requirements	01-dic	18-dic	13
Altium Design	08-ene	15-feb	27
Firmware Implementation	01-feb	05-jul	111
System Analysis			
New Design Requirements	19-feb	28-feb	6
Electronic Devices Election	01-mar	19-abr	36
PCBs Sizes Election	08-mar	10-mar	3
System Design			
SCH Design	01-abr	15-may	32
Firmware Implementation	12-may	08-jul	42
PCB Design	17-may	27-may	9
Can Design	16-jul	18-jul	1
Validation & Documentation			
Circuit's Test	17-jun	08-jul	16
Requirements Verification	01-ago	07-ago	5
Bachelor Thesis Documentation	01-jun	05-sep	69
Defense Preparation	05-sep	13-sep	6



3.5 Power Budget

The power budget needed for developing this device is going to be detailed in Appendix A. On that budget we have taken into account not only the price of the modules but also the price of the plastic for the ”can”, the software and the human resource cost.

3

Chapter 4

System Design

After the different stages of analysis and requirements definition performed in former chapters, this forth chapter will address the **system design**. It will be structured as Section 2.1, so that the **GranaSAT** design structure coincides as much as possible with the **Open CanSat** design. Even though when we talk about a system design we focus on the physical implementation, we will be including in this chapter also the software design since we think is as important as the circuits (the modules won't work unless we tell them what to do). Ergo this chapter's structure will be the same as the Chapter 2's structure except for the last subsection (Section 2.6) which will be moving to Chapter 5.

Before embarking on the system design, we need to remember that during this chapter we intend to address the fifth step of the reverse engineering process described in the Section 1.4. Consequently, with this chapter we will be finishing the reverse engineering process.

4.1 Electronic Design

4.1.1 Schematics Design

Analogue to how we proceeded for building the Czech's engineers design in subsubsection 2.1.1.1, we will be developing several projects (one for each board) in Altium. Just as we mentioned in that section, this program makes the PCB design a relatively straightforward process.

In Subsection 3.3.1, we came into the conclusion that the best PCB's shapes for our design is implementing circular PCBs for the boards that will be introduced in the can, and a rectangular board for the ground board.

We are doing a reverse engineering process to the Open CanSat kit, therefore we will be maintaining the aspects we think are best for our device to operate with maximum efficiency and changing the ones which will give us some kind of trouble.

In consequence, we will be maintaining their three boards structure for the transmitter: one PCB including the main modules, one PCB for the power supply and one last PCB of PTH for extra modules. Note that like we mentioned in Subsection 3.3.2, the GPS antenna must be placed in the outer PCB so that we can communicate freely with the satellites.

Akin to how we did in subsubsection 2.1.1.1, in the SCH we do not only include the circuit, but also we added some information about the modules. In consequence, we have a rather extensive SCH files that we will be including in Section B.2 for not interfering with the reading.

While developing this device, we tested some of the modules we where going to use, such as the BME280 module. This way, we gained confidence and making sure they function correctly. Given that we have a configuration of this microprocessor that already worked, we only need to implement the same circuit. To do so, we would need to make a reverse engineering process to the nodeMCU **ESP-32S** module. We were able to find the SCH [70] for this module online, so we would only have to copy them.

Without any further ado, we are going to start the circuit's explanation. To do so, we will be explaining them by the order of appearance in the SCH.

4.1.1.1 Main Board Hardware

This board will hold all the needed components for the CanSat transmitter modules to work (with the exception of the power supply circuit) as we already discuss in Subsection 4.1.1. Lodging all the modules on the main board will make the PCB design somewhat compact and will not allow to accommodate any other module. Thus, we will avoid that, in case the user wants to add any more modules, these devices will no take the risk of being damaged by the malpractice of using the soldering iron.

Nonetheless, as the GPS antenna we have chosen is bigger than the one they have chosen (using a wire to be connected), we will be placing it on the power board PCB. Even though, as the antenna cable has a sub-miniature version A (SMA) male connector on its end, we will be placing a SMA female connector in the main board so that we can put the cable through a hole and connect it into the corresponding connector. This way, the connections for the main modules will remain on this board.

The SCH implemented for this board is exhibit in Subsection B.2.1.

4.1.1.1.1 Microprocessor

We already discus in Subsection 3.2.1 the microprocessor we are going to be using and the reasons for this election.

The **ESP-32** is a series of low-cost, low-power microcontrollers with integrated WiFi and dual-mode Bluetooth. Depending on the vendor, we could be choosing a wide variety of them based on the memory or antenna we would like [66]. As for us, the best option is to choose the **ESP-32-WROOM-32D** (Figure 4.1) just because is the one used on the nodeMCU **ESP-32S** module we will be using for testing.

(a) *Front view* **(b)** *Back view*

Figure 4.1 – *ESP-32-WROOM-32D microprocessor [65]*

For implementing the circuit for the microprocessor, we followed the datasheet suggestions [65].

The main characteristics we are going to highlight from this microprocessor are:

	ESP-32-WROOM-32D
CPU	Xtensa LX6 32 bits Dual core from 80 MHz to 240 MhZ
SRAM	520 kB (8 kB SRAM in RTC)
FLASH	4 MB (larger available)
ROM	448 kB
POWER SUPPLY	3.0 to 3.6 v

Continued on next page

4

	ESP-32-WROOM-32D
CURRENT CONSUMPTION	Minimum current required to be delivered: 500 mA Average Current Consumption:80 mA Sleep: 2.5 µA (10 µA RTC + RTC memory) Low Consumption: >150 µA
TEMPERATURE RANGE	−40 to 85 °C
WiFi	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 150 Mbps
WiFi MODES	Station / SOFTAP / SOFTAP + STATION / P2P
NETWROK PROTOCOL	Ipv4 / Ipv6 / SSL/TLS / TCP / UDP / HTTP / FTP / MQTT
BLUETOOTH	V4.2 BR/EDR+BLE
HW ENCRYPTION	Yes
DAC	2 DAC Channels of 8 bit
ADC	18 ADC Channels of 12 bits
INTERFACES	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / DAC / TOUCH / PWM / LED
TOUCH SENSOR	Yes (8 Channels)
TEMPERATURE SENSOR	Yes
HALL EFFECT SENSOR	Yes
DIMENSIONS	18 x 25.5 x 3.1 mm^3 8 g
PRICE	2.20 € https://www.mouser.es/

Table 4.1 – *Main Board Modules Description*

Analyzing the table above, we can notice that not only the dimensions and weight are the right fit for our device but also the power supply and the variety of extra features such as the WiFi and the bluetooth.

One last thing we must point out about this table is that this microprocessor allow us to implement the following interfaces:

- 3 different SPI communication lines
- 2 I2C interfaces
- 3 distinct UART connections
- 16 PWM output channels

- 10 Capacitive sensing GPIOs
- 2 I2S interfaces
- 2 DAC Channels of 8 bits
- 18 ADC Channels of 12 bits

Moreover, we need to take into account that the pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on the module and are not recommended for other uses [65].

Once the main characteristic of this module have been pointed out, we can proceed to explain the circuit implemented. For carrying out this circuit, we have followed the microprocessor suggestions [65]. The circuit the datasheet suggest to use this microprocessor is the one showed in Figure 4.2.

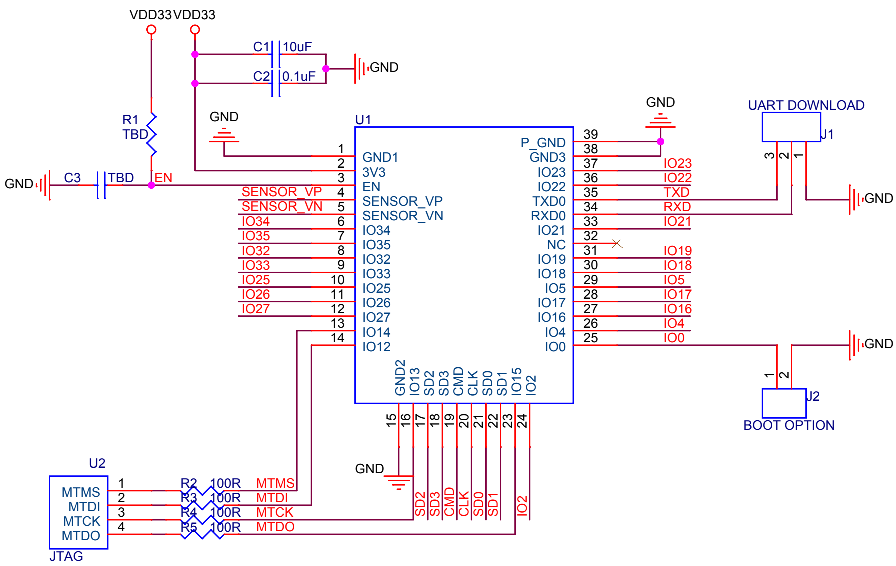


Figure 4.2 – *ESP-32-WROOM-32D microprocessor application circuit [65]*

It is a rather simple circuit in which we will only need to add some capacitor and a resistor on the enable and on the power supply line.

For the enable line, we will need to add the following values:

$$R = 10\ \text{k}\Omega$$

$$C = 1\ \mu\text{F}$$

Finally, on the power supply line even though they have suggested only two ceramic capacitors, we are going to use a 100 µF electrolytic capacitor and a 22 pF ceramic capacitor. The reason for using the electrolytic capacitor is that sometimes, on the nodeMCU it is needed to filtrate the current peaks. As for the ceramic capacitor, we have applied the next formula:

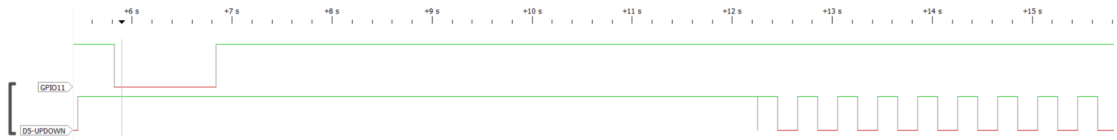
$$C = \frac{I}{\Delta V \cdot f} \tag{4.1.1}$$

The values used for this formula are extracted from the datasheet [65] and are: I = 12 mA, f = 80 MHz and ΔV = 3.3 V. Those values will tell us we need a 45.45 pF ceramic capacitor but as the most proximate available on the laboratory stock is 22 pF, we will be using this one.

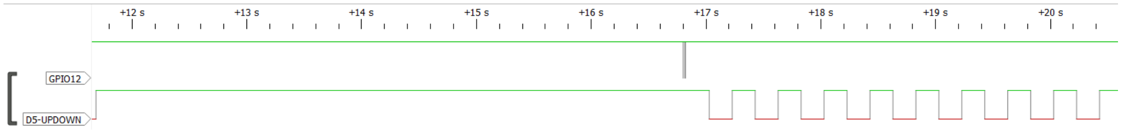
4

From the pin assignment we have implemented, we need to mention that, even though on the datasheet [65] has been said that we should not be using the SPI Flash pins (GPIO6 to GPIO11) for other uses, we have finally used them mainly for LEDs purposes. As we were short in pins, we felt the need of using them. Before doing so, we demonstrated that after uploading the program into the board, this signals where LOW and ready to be used. Demonstrating that this signals were ready to be used in the ESP-32S NodeMCU board is as simple as uploading any program (in our case we used the blinking LED program), and observing how this signals are affected thanks to a logic analyzer device (Figure 4.3). The results for this signals are presented in Figure 4.4.

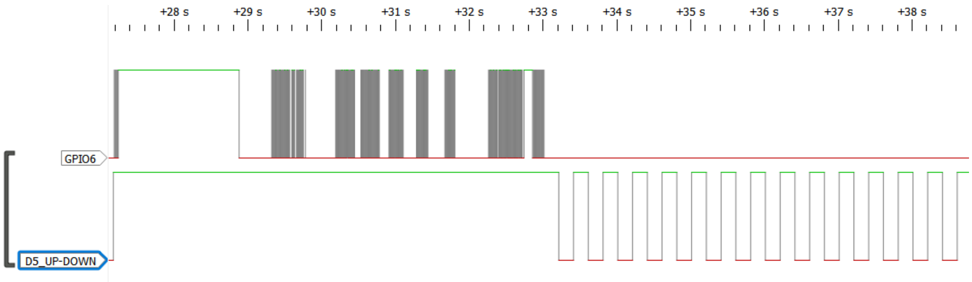
Figure 4.3 – *Logic Analyzer [47]*



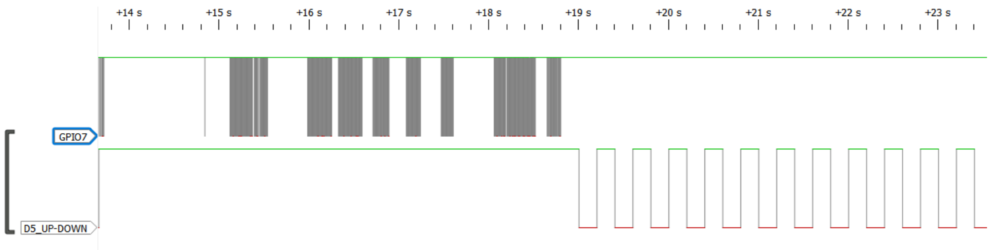
(a) *GPIO11 Signal*



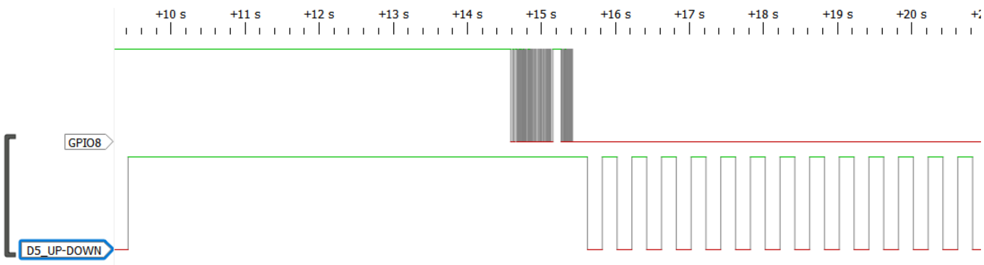
(b) *GPIO12 Signal*



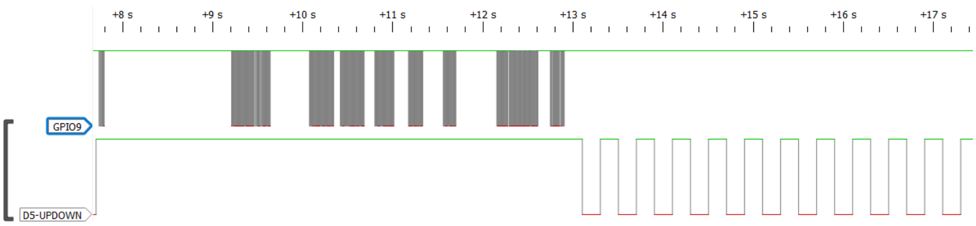
(c) *GPIO6 Signal*



(d) *GPIO7 Signal*



(e) *GPIO8 Signal*



(f) *GPIO9 Signal*

Figure 4.4 – *SPI signals (GPIO6 to GPIO12) during and after uploading a program*

From the images in Figure 4.4 we can conclude that, after uploading the program, these signals flatten out and could be ideally used.

To conclude this module's explanation, we need to think about the fact that we can corrupt the bootloader. It is indeed a very strange situation since this microprocessor protect the registers of the memory Flash so that they can not be accessed by accident. We will refer to Section D.1 for knowing the process for restoring it in case this happens.

4.1.1.1.2 Temperature, Humidity and Pressure Sensor

Following the main requirements that our device must met in order to accomplish the primary mission, is to measure at least the air pressure and temperature. Moreover, since the Czech engineers have introduced a sensor that also measures the air humidity and the device altitude, we have also introduced a sensor that is able to measure those four parameters.

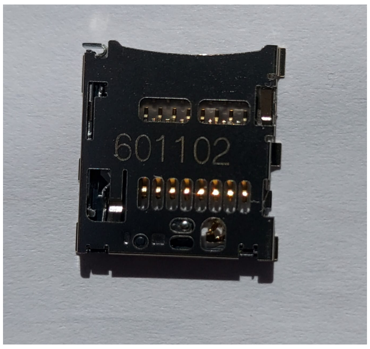
Given that in Subsection 3.2.2 we selected the same module as the Czech engineers for measuring the air parameters,the **BME280**, we will stick to the same circuit as well. Thus, the characteristics and circuit implemented for this module is already explained in paragraph 2.1.1.1.2.

Even though as this sensor has a I2C interface that as we have studied requires some pull-up resistors that depends on the total capacitance on the bus and the frequency, since we plan to route this two tracks as close as possible to the microprocessor, we will omit them as they should not be that important in this case. The reason for taking this decision is mainly that the Czech engineers have not included them either.

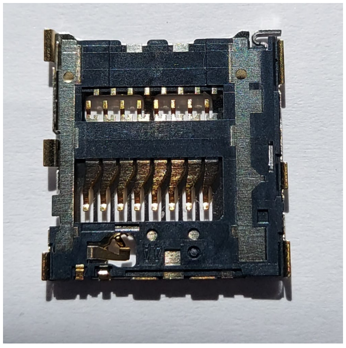
4.1.1.1.3 MicroSD Card Socket

Another critical module for our device is the microSD in order to save the data measured in case the radio communication does not work as expected. One may ask why not save all this data in the microprocessor Flash Memory since we already studied in Table 4.1 this memory has a capacity of 4 MB. There are two main reasons why that is not the best idea: there a write limitation for writing into it and when this memory is overflow it tends to overwrite the first thing we wrote, therefor we would be taking the risk of loosing everything.

Taking that into account, we come across the need of choosing the best SD socket for our device. As we already discuss in Subsection 3.2.3, the best option is the **MOLEX 504528** (Figure 4.5).



(a) *Front view*



(b) *Back view*

Figure 4.5 – *MOLEX* **504528** *socket [69]*

The only thing we must point out of this module is it is a 8 pin module requiring a SPI interface. It comes also with a detection switch pin that is leveled to GND when a card is inserted.

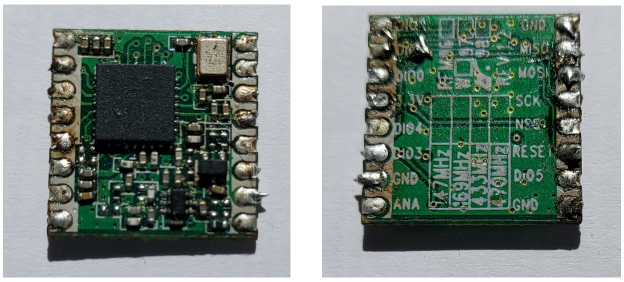
The circuit implemented for this module is as simpler as connecting the pins as shown in Table 4.2. Note that we have included a 1 µF bypass capacitor and a 4.7 nF decoupling capacitor. For choosing the electrolytic capacitor we have only observe the electrolytic capacitor frequency response graphic and choose according the one that will resist a 25 MHz frequency that this modules implements. On the contrary, for choosing the ceramic capacitor value we have resort to Equation 4.1.1 where I = 0.5 A, V = 3.3 V and f = 25 MHz [69].

MicroSD Module Pins	ESP-32-WROOM-32D Pins
DATA2	Not Connected
CD/DATA3	SPI microSD CS (GPIO25)
CMD	SPI MOSI (HSPI)
VDD	+ 3.3 V power supply
CLK	SPI SCLK (HSPI
VSS	GND
DATO0	SPI MISO (HSPI)
DATO1	Not Connected
CARD DETECT	GPIO2

Table 4.2 – *MicroSD Module interface to ESP-32-WROOM-32D*

4.1.1.1.4 Radio Transmitter Module

The last module needed for achieving the primary mission is the radio module. We already saw in Subsection 3.2.4 that the best module for our device is the **RFM69** radio module (Figure 4.6) not only for its dimensions but also for giving us the possibility to communicate with a 2000 m distance.



(a) *Top view* **(b)** *Bottom view*

Figure 4.6 – ***RFM69 radio module*** **[71]**

The main features of this module that we should know are exhibit in Table 4.3.

	RFM69
POWER SUPPLY	1.8 to 3.7 V
CURRENT SUPPLY	Receive mode : < 12.1 mA Transmit mode : < 120 mA Sleep mode : < 1 µA Standby mode : < 1.5 µA
FUCTIONING FREQUENCY	433 MHz
DISTANCE	2000 m to 20 km depending on the antenna
TEMPERATURE RANGE	– 40 to 85 ºC
SENSITIVITY	– 126 dBm
MODULATION	FSK, GFSK, MSK, GMSK , OOK
OTA DATA RANGE	100 bps to 1 Mbps
LINK BUDGET	– 148 dBm
INTERFACE	Serial SPI
TEMPERATURE SENSOR	Yes
LOW BATTERY DETECTOR	Yes
DIMENSIONS	16 x 16 x 2 mm^3 0.949 g

Continued on next page

	RFM69
PRICE	3.26 € https://es.aliexpress.com/

Table 4.3 – *RFM69 main features [71]*

One thing we need to pay attention to is that this module only allow us to implement the 433 frequency range, therefore we are more limited in case multiple device are transmitting on this frequency range. Even though we have not included this feature in Table 4.3, we must know than the **frequency bandwidth** is 7.8 – 500 kHz. Also we must know that the **spreading factor** (SF) can be set between 6 – 12 (larger SF increases the OTA reducing the data rate and improving the communication range).

The last thing we must highlight from this module before entering to discus the circuit implementation is the packet structure. This structure is the one exhibit in Figure 4.7 where the preamble length vary from 6 to 65535, the header depends on the mode of operation (*ImplictHeaderMode* or *RegSymbTimeoutMsb*) and the payload packet is a variable-length field with a FIFO structure.

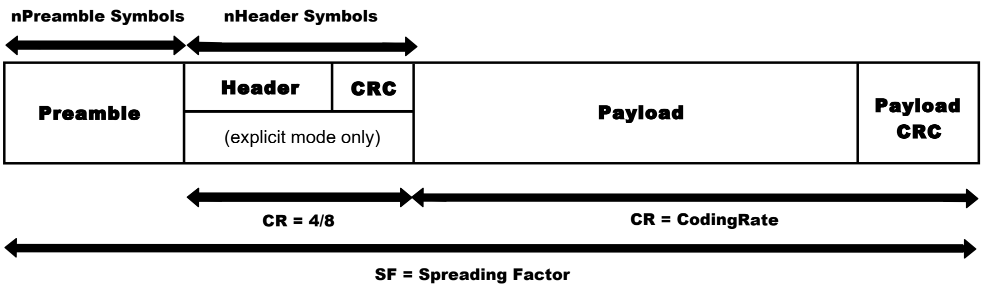


Figure 4.7 – *RFM69 Packet Structure [71]*

Once all the module's characteristics have been studied, we can proceed to explain the circuit implementation. This module requires a SPI interface so we will be interfacing this module as shown in Table 4.4. For interfacing this module we have followed the datasheet recommendations [71].

RFM69	CONNECTION
GND	Board's GND
MISO	Microprocessor HSPI MISO pin
MOSI	Microprocessor HSPI MOSI pin
SCLK	Microprocessor HSPI SCLK pin
NSS	Microprocessor CS pin (GPIO15)
RESET	GPIO19 (Active Low for 100 µs)
DIO5	Not Connected

Continued on next page

RFM69	CONNECTION
GND	Board's GND
DIO2	Not Connected
DIO1	Not Connected
DIO0	GPIO23
3V3	+ 3.3 V power supply
DIO4	LED Radio
DIO3	Not Connected
GND	Board's GND
ANT	17.32 cm wire

Table 4.4 – ***RFM69*** *interface [71]*

4

The datasheet also suggests connecting the DIO0 pin for software configuration and also adding a 10 µF bypass capacitor and a 0.1 µF decoupling capacitor for filtering the possible noise. Note that there should be no problem sharing the HSPI lines with the microSD if the CS pin is different.

The LED connected to the DIO4 pin is for indicating when we are transmitting anything via radio. The rest of the DIO pins can be left floating due to their internal pull-ups.

The last thing we need to mention is the radio antenna. Analogue to what the Czech engineers decided, we have implemented a simple electronic cable to act as an antenna. This is a monopole antenna ($\frac{\lambda}{4}$) which need a length of 17.32 cm according to Equation 2.1.1.

4.1.1.1.5 GPS Receiver Module

One of the secondary mission the Czech engineers introduced is to receive the CanSat location from the GPS. Like we already mentioned in paragraph 2.1.1.1.5 this is the best way for locating the device in case the landing is not where is supposed to be.

According to Subsection 3.2.5, the most appropriate module for our application is the **ATGM336H** (Figure 4.8). The main features that we have to be aware of from this module are presented in Table 4.5.

Figure 4.8 – *ATGM336H module* [33]

	ATGM336H
POWER SUPPLY	2.7 to 3.6 V
POWER CONSUMPTION	< 25 mA
TEMPERATURE RANGE	− 40 to 85 ºC
UPDATE RATE	1 Hz to 10 Hz
ACCURACY	Horizontal position: < 2 m Velocity : < 0.1 m/s Time : < 30 ns
INTERFACES	UART1/UART2
SENSITIVITY	Navigation : -162 dBm Tracking : -162 dBm Reacquisition : -162 dBm
DYNAMIC PERFORMANCE	Maximum Altitude: 18000 m Maximum Velocity : 515 m/s Maximum Acceleration: < 4 G
DIMENSIONS	10.1 x 9.7 x 2.4 mm^3 0.612 g
PRICE	2.77 € https://es.aliexpress.com/

Table 4.5 – *ATGM336H main features* [32]

The only thing we need to highlight from this characteristics is that, on the contrary to their design, the module we have chosen needs to be connected with an UART interface instead of an I2C interface. Another thing we need to consider is the navigation satellite system that can implement. In this case, as the module we have acquire is the ATGM336H-5N-31, it can only be implemented the GPS system and the BeiDou Navigation Satellite System (BDS). The main difference from both of them are the country of origin (United States or China) and the frequency of operation. The different frequencies needed to connect with each system (and the GLONASS from the Soviet Union) are the ones listed in Table 4.6.

SYSTEM	FREQUENCY
GPS	L1: 1575.42 MHz L2: 1227.60 MHz
GLONASS	L1: 1602 MHz L2: 1246 MHz
BDS (BeiDou)	B1C/B1I/B1A : 1575.42 MHz B2a/B2b: 1191.795 MHz B3I/B3Q/B3A: 1268.52 MHz Bs test frequency: 2492.028 MHz

Table 4.6 – *GPS, GLONASS and BDS frequencies*

The circuit we have implemented for this module is the one suggested in the user manual. This circuit is the one shown in Figure 4.9.

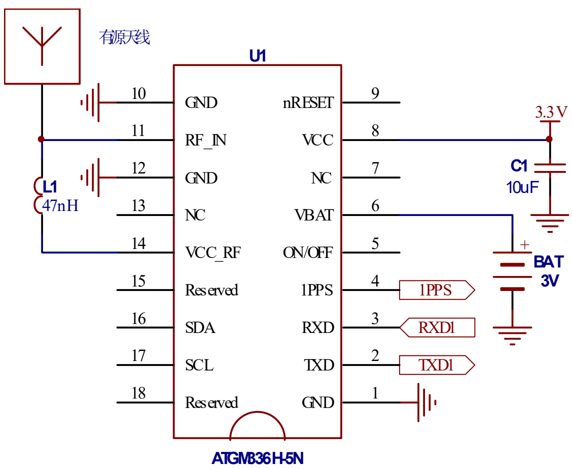


Figure 4.9 – *RFM69 Packet Structure [71]*

It should be noted that for implementing the UART connection, we decided to use a **software UART**.

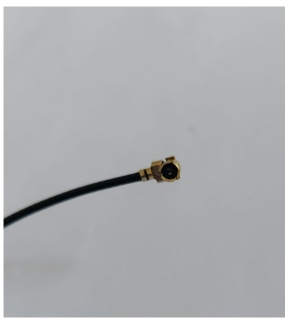
For the antenna we have chosen also a different one than the one they used even though both are ceramic antennas. We will stick to the GPS system, therefore we will need a 1575.42 MHz frequency. The ceramic antenna we have chosen is the one shown in Figure 4.10 and the main features for this module are:

	GPS ACTIVE CERAMIC ANTENNA
POWER SUPPLY	3.0 to 5 V
CURRENT CONSUMPTION	10 mA
FUNCTIONING FREQUENCY	1575.42 $\pm$ 3 MHz
IMPEDANCE	50 Ω
ANTENNA GAIN	28 dBi
INTERFACE	IPX13
WIRE LENGTH	Rf1.13 12 cm
DIMENSION	25 x 25 x 6.5 mm^3 11.2 g
PRICE	2.84 € https://es.aliexpress.com/

Table 4.7 – *GPS CERAMIC ANTENNA MODULE (ACTIVE) datasheet* [13]

(a) *Top view*

(b) *Bottom view*



(c) *Connector detailed*

Figure 4.10 – *Ceramic Antenna*

4.1.1.1.6 LEDs

We will be using some LEDs to visually indicate something. The main things we want to indicate are:

- When all modules are ON
- When the radio module is transmitting

- When the program is being loaded on the board
- When a second has passed (1 pulse per second (1PPS))

Besides those visual indications, we will be implementing two extra LEDs for the user election.

The LEDs we will be using to do so are some **BLUE, RED** and **WHITE LEDs** available on the laboratory stock. Since we do not own their datasheet, we experimentally draw their I-V curve for modeling them and ensuring a good design. The diode experimental curve and the diode modeling process is being explained in Section C.1.

The main features of this LEDs that we should be taking into account for the circuit design are summarized in Table 4.8 (BLUE), Table 4.9 (WHITE) and Table 4.10 (RED).

4

	BLUE LED
FORWARD VOLTAGE	2.47 to 3.737 V
FORWARD CURRENT	0.06 mA to 85.55 mA
TEMPERATURE RANGE	– 30 to 85 ºC
DIMENSION	2 x 1.25 x 0.5 mm^3 (0805) 23 mg

Table 4.8 – *BLUE LEDs characteristics*

	WHITE LED
FORWARD VOLTAGE	2.476 to 3.831 V
FORWARD CURRENT	0.015 mA to 95.374 mA
TEMPERATURE RANGE	– 40 to 85 ºC
DIMENSION	2 x 1.25 x 0.5 mm^3 (0805) 23 mg

Table 4.9 – *WHITE LEDs characteristics*

	RED LED
FORWARD VOLTAGE	1.675 to 3.031 V
FORWARD CURRENT	0.03 mA to 42 mA
TEMPERATURE RANGE	– 30 to 85 ºC
DIMENSION	2 x 1.25 x 0.5 mm^3 (0805) 23 mg

Table 4.10 – *RED LEDs characteristics*

As for the circuit we have implemented for them, we be dividing them into three different ones. Two of them are pretty simple and the third one is a little more complex.

For the radio transmitter function and the ones for the user election the circuit implemented is an **active low** resistor-LED circuit as exhibit in Figure 4.11.

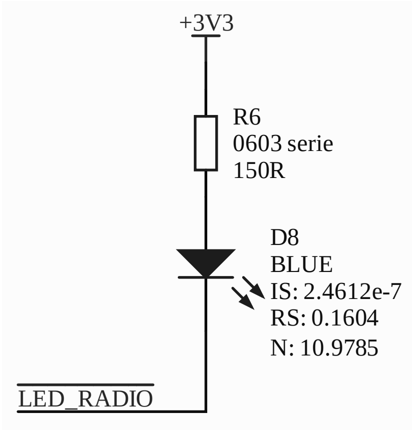


Figure 4.11 – *Active-Low LED circuit (I)*

For the 1PPS LED and for indicating that the modules are ON, we will be implementing the same resistor-LED circuit but instead of activating it when the pin is LOW, we will be activating it when the pin is HIGH. To do so, the circuit needed is as presented in Figure 4.12.

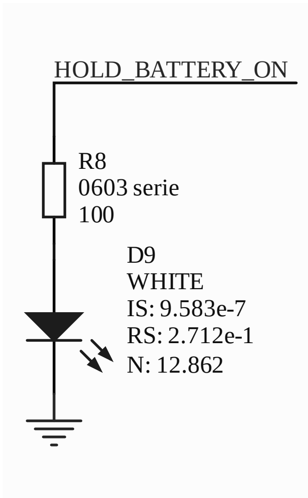


Figure 4.12 – *Active-High LED circuit (II)*

The last LED circuit implemented is for indicating when the program is being uploaded. To do so, we will be implementing a double color LED: when one color is lit, the other is off and viceversa. This last circuit is as presented in Figure 4.13.

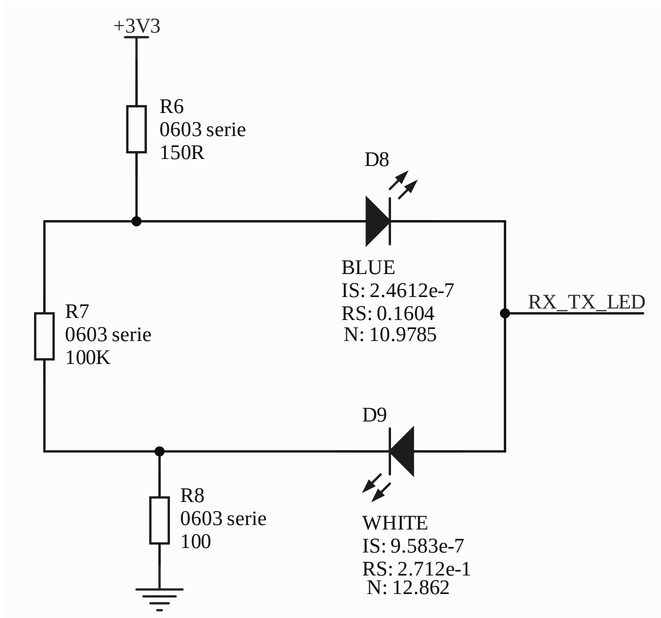


Figure 4.13 – *Duble-implementation LED circuit (III)*

4

Note that the resistor depends on the LED forward current and voltage, therefore, for the Figure 4.11 and Figure 4.12 circuits, the resistor chosen are:

$$R = 150\text{R}\ \Omega\ \text{(BLUE)}$$
$$R = 150\text{R}\ \Omega\ \text{(WHITE)}$$
$$R = 800\text{R}\ \Omega\ \text{(RED)}$$

4.1.1.1.7 Buzzer

One important module that the Czech engineers did not implemented and we think is an important one is a **buzzer**. The main reason why we think this module is important is for the CanSat recovery. If we activate some kind of noise, if the CanSat lands on some bushes or something like that, we will be able to localize it from the sound it emits.

The buzzer we have selected is the **KXG1205** buzzer (Figure 4.14). We have chosen this particular module just because it was available in the laboratory stock. The main characteristics of this module are exhibited in Table 4.11

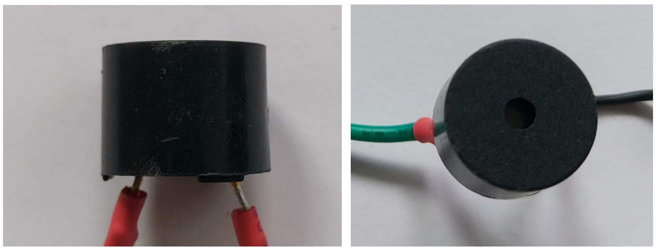


Figure 4.14 – ***KXG1205*** *buzzer module*

	KXG1205 BUZZER
SUPPLY VOLTAGE	1.675 to 3.031 V
CURRENT CONSUMPTION	0.03 mA to 42 mA
COIL RESISTANCE	47.0 $\pm$ 7.0 Ω
SPL	92 dB (min: 85 dB)
FREQUENCY	2.4 kHz
TEMPERATURE RANGE	$-$ 30 to 70 °C
DIMENSION	12 x 9.5 x 0.5 mm^2 1.6 g

Table 4.11 – ***KXG1205 BUZZER*** *characteristics [15]*

The main features to highlight from this module are the frequency of operation, which we will need to take into account when programing it and the current consumption. Also, we have to be aware of the size of this module, which is one of the most voluminous modules.

As we have mentioned, the current consumption is an important parameter to be aware of. If the current consumption is too high, the microprocessor pin may not provide enough current for supplying this module. Also, it could interfere with other modules if all are working at the same time. Keeping this in mind, the circuit we have implemented is a simple one since we have tested that supplying the buzzer directly from the pin is enough for it to work. Therefore, we will only need to connect the positive pin of the buzzer to the microprocessor and the negative pin to GND as shown in Subsection B.2.1.

4.1.1.1.8 Current \Power Monitor

Like we mentioned in Subsection 3.2.7, even though this device is intended for electronic beginners who are not too aware of what is the consumption or why it should be taken into account when designing any device, it is good idea to introduce a device able to tell us such information to bring them closer to those topics. If the module is implemented, they will investigate more in this issues.

Therefore, to do so we have chosen the same module the Czech engineers used, the **INA219** (Figure 4.15).

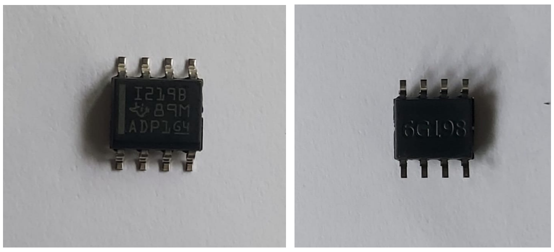


Figure 4.15 – *INA219 module*

Hence, the characteristics that we exhibited in paragraph 2.1.1.1.7 are the same and we stick to the things we highlight then. We only need to remark that this module is available with two different packages, they have chosen the smaller one, the **SOT-23** package, and we have chosen the bigger one, the **SOIC-8**, since we will solder it manually and it will be more comfortable for us. The sizes for those two modules are:

PACKAGE	BODY SIZE (NOM)
SOIC-8	3.91 mm x 4.90 mm
SOT-23	1.63 mm x 2.90 mm

Table 4.12 – *INA219 available packages [45]*

As for the circuit we have implemented, given that we have the same module, is the same circuit they have implemented. They have followed the datasheet [45] suggestions for this circuit and so did we. The only thing we have changed from their implementation is the shunt resistor. We have chosen a 0.2 Ω 1206 SMD resistor.

As for the circuit we have implemented, given that we have the same module, is the same circuit they have implemented. They have followed the datasheet [45] suggestions for this circuit and so did we. The only thing we have changed from their implementation is the shunt resistor. We have chosen a 0.2 Ω 1206 SMD resistor. For choosing this resistor, we have implemented the following formulas.

$$V_{ADC} = \frac{\mathrm{V_{DD}} - V_{SS}}{2^{ADC}} \tag{4.1.2}$$

If our ADC consists of 12 bits and the $\mathrm{V_{DD}}$and$\mathrm{V_{SS}}$ are the $\mathrm{V_{DD}} = 3.3$ V, the ADC voltage resolution would be:

$$V_{ADC} = 805.66\ \mu\mathrm{V}$$

Moreover, if we choose a PGA of 8 bits, according to its datasheet [], the maximum voltage sense through the shunt resistor we can measure is $\mathbf{V_{SHUNT}} = \pm$ **320 mV**. With that sensing voltage, the voltage through the shunt resistor can be:

$$3.7\ V - 0.32\ V = 3.38\ V.\ \text{As}\ 3.38\ V > 3.3\ V$$

Therefore, we have picked the perfect PGA for our device.

The last thing we need to know for getting the shunt resistor value is to know the maximum current. This current would be the current consumption of the device, which is calculated in Subsection 4.1.3.

If we assum a maximum current value of 1.5 A (a value between the typical consumption and the maximum one), we will only need to apply the Ohm Law as presented in Equation 4.1.3.

$$R_{SHUNT} = \frac{V_{SHUNT}}{I_{MAX}} \tag{4.1.3}$$

If $V_{SHUNT} = 320$ mV and $I_{MAX} = 1.5$ A the shunt resistor value would be:

$$\mathbf{R_{SHUNT} = 0.2\ \Omega}$$

To conclude, we also have to mention that, analogue to what we mentioned in paragraph 4.1.1.1.2, even though this device is interface with I2C that requires some pull-up resistor, we have omitted them for two reasons. The first one is that we intend to make those tracks as short as possible and the second one is that as the Czech engineers did not add them either, we will stick to saving as much space as possible.

4.1.1.1.9 Hardware Reset

As we mentioned in Subsection 4.1.1, when developing the new product SCH, we tested some of the modules on the ESP-32S. While doing that, we decided to make a reverse engineering process to this board as it was using the same microprocessor we intended to use. From this process we learned how to implement the system for doing the **reset** and **booting** of this microprocessor.

Both are really simple circuits that depends on detecting a low signal on its pins.

4.1.1.1.9.1 Reset

For resetting the microprocessor, the datasheet [65] indicates that we should put the *enable/reset* pin (third pin) LOW for at least 50 µs. Consequently, as this time is a very short time, for ensuring the power supply during the power-up, we added a RC circuit (R = 10 kΩ, C = 1 µF).

As for the reset, we only need a simple circuit that consists of a pull-up resistor and a button. When the button is pressed, the *Enable* pin is at GND level. When the button is not pressed, this button is at 3.3 V. Therefore, we only need to be sure to press the button for at least 50 μs to be able to perform the module reset.

4.1.1.1.9.2 Boot

In the microprocessor datasheet [65], indicates that depending on the GPIO0 input, we can enter the serial bootloader or run the program in the flash. For entering the ROM serial bootloader for esptool.py we only need to hold the GPIO0 pin LOW on reset. Otherwise, we should pull this pin high.

For implementing this circuit we will proceed as in for the reset. We will be designing a circuit with a pull-up resistor and a button. Consequently, in normal mode this pin will be held up with the pull-up resistor. However, when the button is pressed, we will enter in the ROM serial bootloader mode.

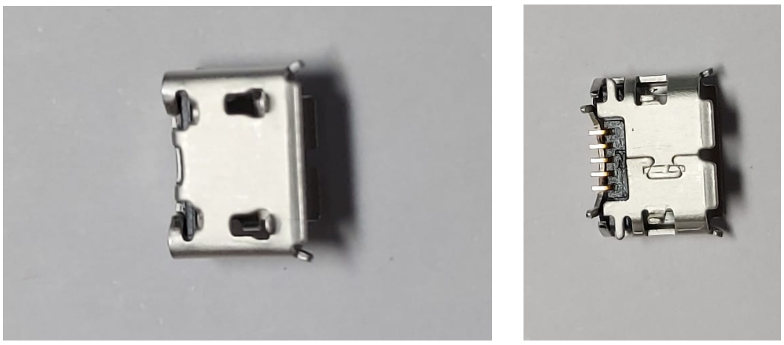
4

4.1.1.1.10 USB + Power Management

On this section we will be including the circuits that we have implemented for the USB, for doing the USB to serial conversion and the circuit needed for converting the 3.7 V tension given by the battery to the 3.3 V tension needed for supplying all modules. Moreover, we will include the circuit needed for programming the microprocessor.

4.1.1.1.10.1 USB

For the USB we have implemented the same circuit as them, even though we have chosen a different module as we discus in Subsection 3.2.9. As both of them are type-B USB, the pins are the same and therefore it is needed the same circuit. The USB we will be using is the $\mathbf{MOLEX47346-0001}$ (Figure 4.16).



(a) *Front View* **(b)** *Back View*

Figure 4.16 – ***MOLEX47346*** **– 0001** *module [49]*

The main characteristics that our USB has are exhibited in Table 4.13.

	MOLEX47346 – 0001
SUPPLY VOLTAGE	30 V AC maximum
CURRENT SUPPLY	1.8 A maximum per contact
TEMPERATURE RANGE	– 30 to 85 °C
USB DEVICE INTERFACE	2.0
DIMENSION	7.80 x 5.0 x 2.94 mm^3 16.7 mg
PRICE	0.83 € https://es.rs-online.com/

Table 4.13 – ***MOLEX47346*** **– 0001** *main characteristics [49]*

Analogue to what they did, for protecting the device from a possible ESD, we have introduced an ESD protector between the USB and the USB to serial communication chip.

The ESD module we have chosen is not the same one as they had selected, but has similar values. The ESD chosen is the **HDMIULC6-4SC6** (Figure 4.17) since the one they used is not available. The main characteristics of this module are listed in Table 4.14.

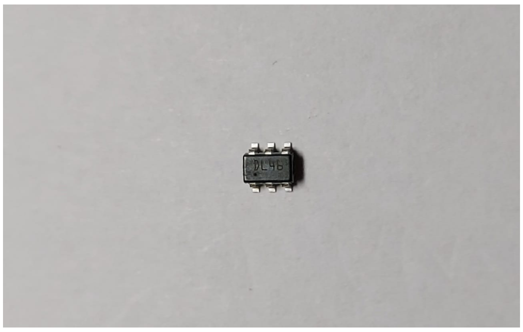


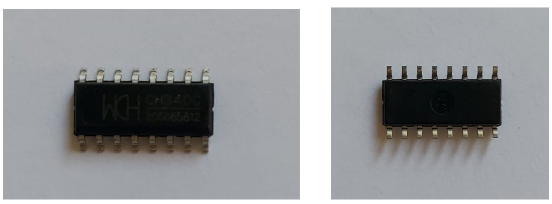
Figure 4.17 – ***HDMIULC6-4SC6*** *module [41]*

	HDMIULC6-4SC6
VOLTAGE CLAMP	17 V
BREAKDOWN VOLTAGE	6 V
LEAKEAGE CURRENT	0.5 µA
DIMENSION	3.05 x 1.75 x 1.3 mm^3 16.7 mg
PRICE	0.55 € https://es.rs-online.com/

Table 4.14 – ***HDMIULC6-4SC6*** *main features [41]*

4.1.1.1.10.2 USB to Serial

If we remember from the Chapter 2, the microprocessor they used was able to convert the USB to serial communication for programming the device through the USB. In our microprocessor, this conversion is not automatically made and we will need an extra module to do so. The module we have chosen is the **CH340C** (Figure 4.18) since is the one available in the laboratory stock. Following the reverse engineering process we are doing to the ESP32S nodeMCU board [70], we observe that they also have included a chip to do such conversion but instead they have used the CP2102 chip.



(a) *Top view* **(b)** *Bottom detailed*

Figure 4.18 – *CH340C chip [36]*

The main features we will need to be aware of when designing the circuit for this device are listed in Table 4.15.

	CH340C
SUPPLY VOLTAGE	3.0 V to 3.3 V
CURRENT SUPPLY	7 mA (20 mA max)
TEMPERATURE RANGE	− 20 to 70 °C
COMMUNICATION BAUD RATE	50 bps to 2 Mbps
USB DEVICE INTERFACE	2.0
DIMENSION	3.9 x 1.27 mm^2 16.7 mg
PRICE	0.68 € https://es.aliexpress.com/

Table 4.15 – *CH340C main features [36]*

This chip is a rather sensitive module, so we will need to be very careful to not supplying it over their supplying needs. Another important thing we have to remark from this module is that it has an internal crystal oscillator so we will not need to add an external one.

The circuit we have implemented for this chip is as suggested in the datasheet. This circuit only requires a bypass capacitor of 0.1 µF

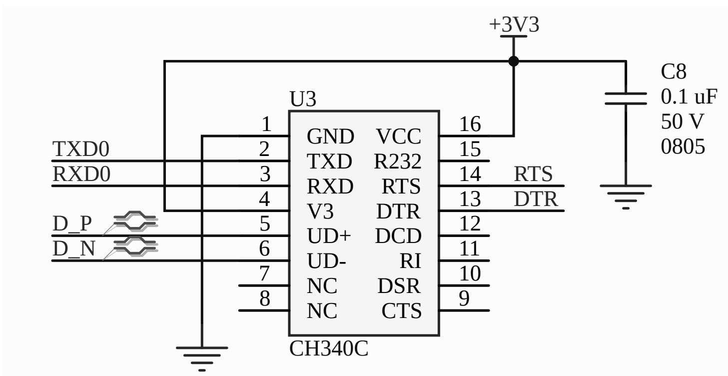


Figure 4.19 – *CH340C implemented circuit [36]*

The only thing we have to highlight form this circuit is that this module is the one connecting the USB and the microprocessor. In fact, this module is the one needed for producing the signals needed for programming our device.

4.1.1.1.10.3 Programming Circuit

Going back to the part where we said that we are doing a reverse engineering process to the ESP-32S nodeMCU board, we will need to go back to the SCH of this board [70] for observing how they have implemented this circuit.

The circuit they have implemented require two bipolar junction transistor (BJT) acting as a electronic switches between the *DTR-RESET* pins and between the *RTS-GPIO*0 pins. For controlling the current rate flow into the BJT gate terminal, they have added a 12 kΩ resistor.

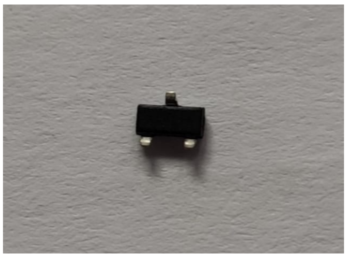


Figure 4.20 – 2N7002 ***NPN MOSFET***

4

Since we are implementing an electronic switch, we will be replacing this BJT with two MOSFETs since they have a higher switching speed. Moreover, since the MOSFETs has a higher gate resistor (less current consumption) we may omit the resistor they have added for controlling the current. However, we can not simply change a circuit that we already know its working without testing if it will still be working as expected with the changes introduced. To do so, we have simulated in altium the circuit with two **2N7002 NPN MOSFET** (Figure 4.20).The signals we will be expecting are shown in Table 4.16, and the one we simulate are the ones exhibit in Figure 4.22. We have also included the experimental signals (Figure 4.21) we measured with the logic analyzer (Figure 4.3) on the ESP-32S board when uploading any program.

AUTO PROGRAM CIRCUIT			
DTR	**RTS**	**RESET**	**GPIO0**
1	1	1	1
0	0	1	1
1	0	0	1
0	1	1	0

Table 4.16 – ***DTR****-RESET/****RTS****-GPIO0 expected signals*

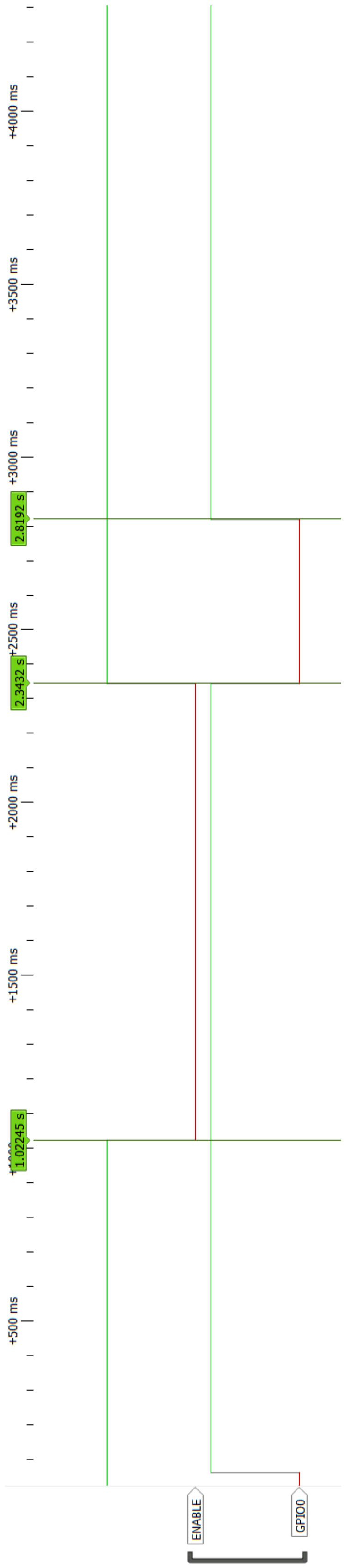
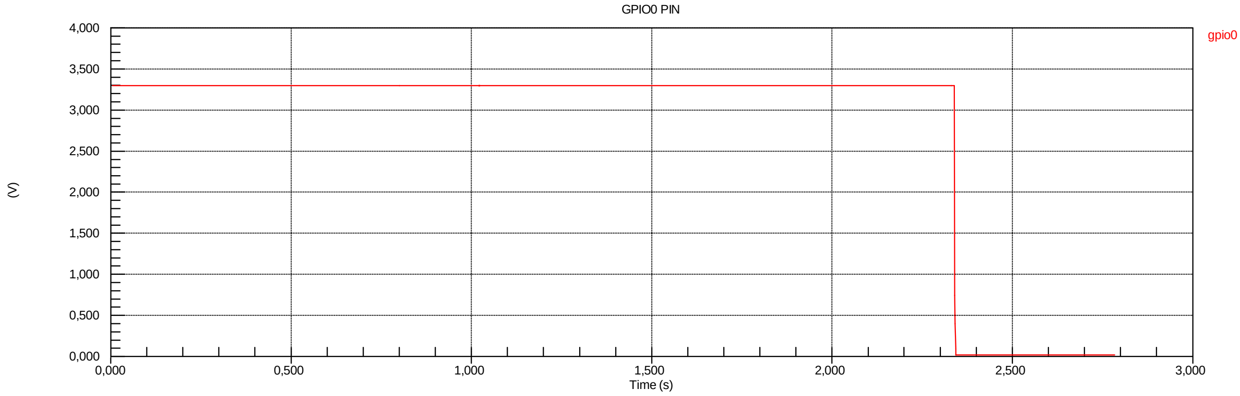
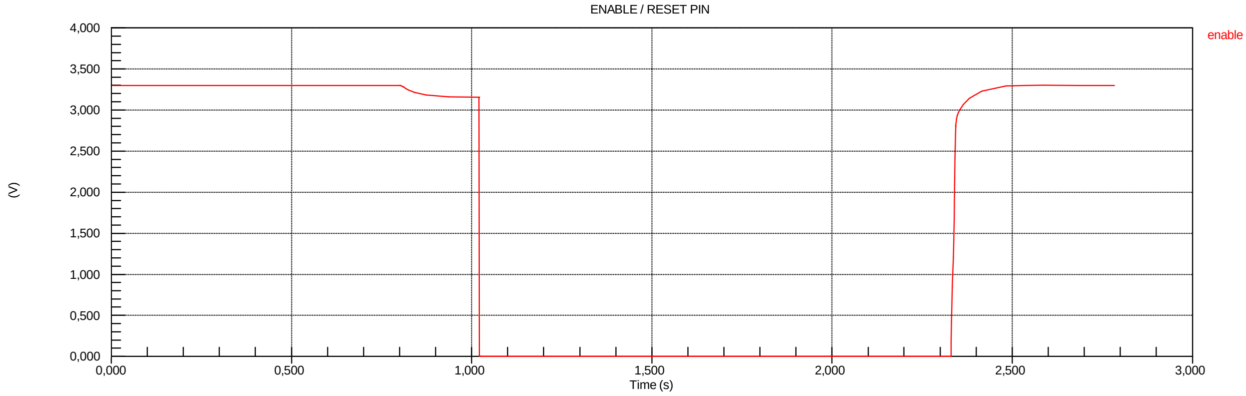


Figure 4.21 – *DTR-RESET/**RTS**-GPIO0 during upload (tested in ESP-32S)*



(a) *GPIO0 Pin*



(b) *Enable/Reset Pin*

Figure 4.22 – *DTR-RESET/**RTS**-GPIO0 simulated in Altium*

For the Altium simulation, we have introduced a pulse width modulation (PWM) source on the DTR and RTS pins so that we can know how the 2N7002 NPN MOSFET would react to such signals. With the Figure 4.22, we can conclude that we can be replacing the BJT with the MOSFET and expect the same behavior saving the space, weight and money that the two extra resistors would add.

Therefore, the circuit we have implemented is the one presented in Figure 4.23.

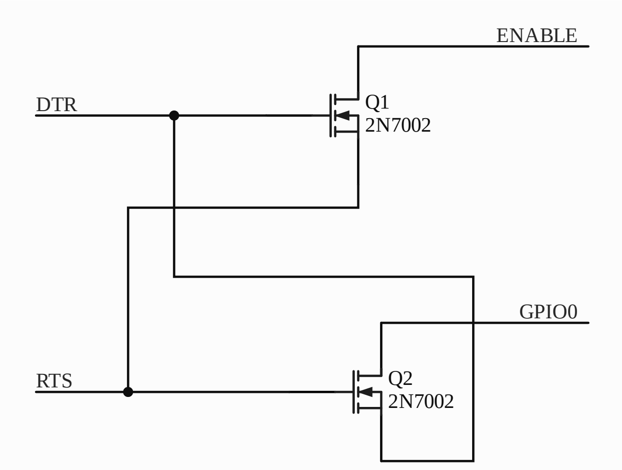


Figure 4.23 – *DTR-RESET/**RTS**-GPIO0 program circuit*

4.1.1.1.10.4 Buck Converter

Even though the Czech engineers added a buck converter and a boost converter to keep two supply lines, one of 3.3 V and another of 5 V, in our design we will only be carrying out one supply line of 3.3 V for all the modules. This way we will optimize the power consumption as well as save the space, weight and budget this circuit would require given that all modules can be supplied with 3.3 V.

Like we discuss in Subsection 3.2.8, we felt the need of changing the buck converter they used for the **AP3429/A** buck converter (Figure 4.24) since we will be requiring more current.

Figure 4.24 – *AP3429/A [31]*

The main features we will need to know about this module are the one exhibit in Table 4.17.

	AP3429 \A
INPUT VOLTAGE	2.5 V to 10 V
INPUT CURRENT	Quiescent Current : 5 mA Shutdown Current < 1 µA

Continued on next page

	AP3429 \A
OUTPUT CURRENT	Up to 2 A
SWITCHING FREQUENCY	1 MHz
TEMPERATURE RANGE	– 40 to 85 ºC
SYNCHRONOUS RECTIFIER	Yes
DIMENSION	2.9 x 1.6 x 0.87 mm^3 16.7 mg
PRICE	0.346 € https://www.mouser.es/

Table 4.17 – *AP3429/A main characteristics [31]*

As for the circuit we have implemented for this module, is the one suggested by the datasheet (Figure 4.25).

4

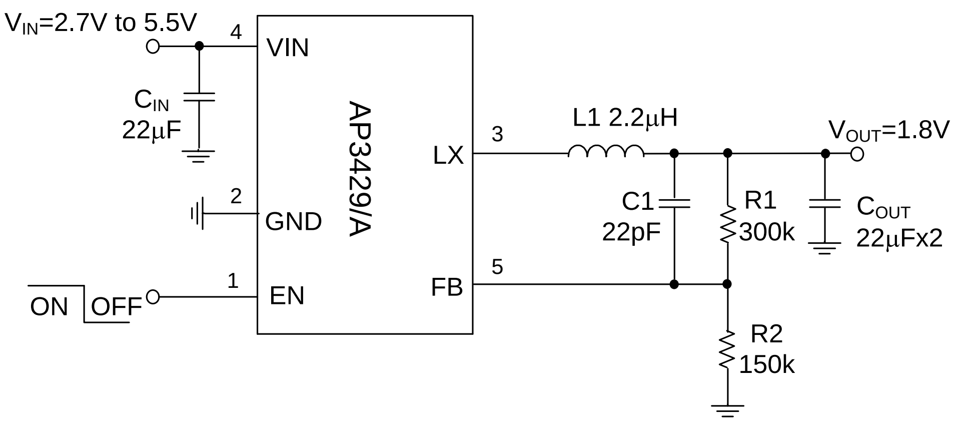


Figure 4.25 – *AP3429/A implemented circuit [31]*

There are two things we must take into account when implementing this circuit. The first one is that in order to assure the maximum current (2 A), the inductor we should use needs to be able to tolerate a maximum current above the 2 A we will be requiring, therefore we should be taking this parameter into account when buying it. The second thing we must be aware is that the circuit presented in Figure 4.25, the output voltage is 1.8 V instead of 3.3 V we will be requiring. This output voltage depends on the **Feedback Voltage** and on the voltage divider. Consequently, for our device we must change those two resistor that we will be calculating following the voltage divider equation in Equation 4.1.4.

$$V_{FB} = V_{OUT} \cdot \frac{R_2}{R_1 + R_2} \tag{4.1.4}$$

Given that in the datasheet [31] we have extracted the information that the threshold voltage of the feedback pin is 0.6 V, we can find the resistor values as exhibited in Equation 4.1.5

$$R_1 = (1 - \frac{V_{FB}}{V_{OUT}} \cdot R_2 \cdot \frac{V_{FB}}{V_{OUT}}) \tag{4.1.5}$$

Now solving for $V_{FB} = 0.6$ $V_{OUT} = 3.3$ V and assuming $R_2 = 120$ kΩ, the value needed for R_1 will be: $R_1 = 560$ kΩ

4.1.1.1.11 Bus Connector Main Board

The last module we need to mention from this board is the bus connector (Figure 4.26). This module is intended for allowing us to extend this board signals to the Uni Board and for powering the device by bringing to this board the battery tracks.

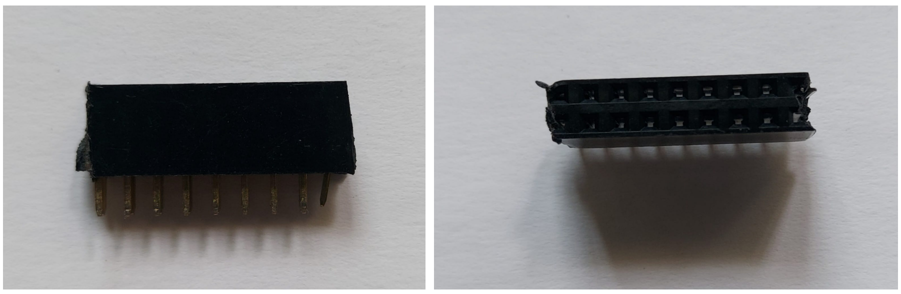


Figure 4.26 – *Bus Connector*

The main characteristics of this board are summarized in Table 4.18.

	Pin Header
TYPE	Female
ROWS	2
PINS	7
PITCH	2.54 mm
INTRA-ROW SPACE	2.54 mm
MOUNTING	THT
DIMENSION	18 x 11.2 mm^2 0.759 g

Table 4.18 – *DTR-RESET/RTS-GPIO0 expected signals*

The signals we are going to be sending from this board to the UniBoard are the ones listed below:

- SDA and SCL line from the I2C interface

- MISO,MOSI and SCLK lines from the SPI interface
- Two free GPIOs pins (to be used for the SPI CS for example)
- GPIO0 pin that even though it it used for programming, it can be used afterward without interfering the normal operation mode
- two GPIOs to be used for a LED indicator
- the 3.3 V line in case the extra modules on the Uni Board needs it
- the GND line for connecting the three boards to the same GND plane
- One last GPIO that is used for configuring the radio module but it could be sacrifice in case some extra module requires it.

The last pin of this module corresponds to the battery line that we will be introducing into this board and into the Uni Board as well.

4.1.1.1.12 Power Switch

4

At this point where we already have introduced all the modules and its circuits, we can wonder how we can turn ON or OFF this device. We have also implemented a circuit to do such thing.

When thinking about how we could be turning ON or OFF our device we run into the necessity of finding some kind of mechanism for preventing the modules to get their power supply. As all modules are supplied by the buck converter we talked about in subparagraph 4.1.1.1.10.4, we need to implement a mechanism to shut down this module when turning the device OFF and powering up when turning it ON.

The perfect way to do so is with the ***Enable pin*** of the AP3429/A module. From the datasheet [31] we have learned that if this pin have a tension above 1.4 V, this module is ON. However, if on this pin is a tension below 0.6 V, this module would shut down. Consequently, we only need to find a mechanism to enable this device with a tension above 1.4 V whenever the device is up and shutting it down with a > 0.6 V tension when the device is off.

The circuit implemented to do so is the one showed in Figure 4.27.

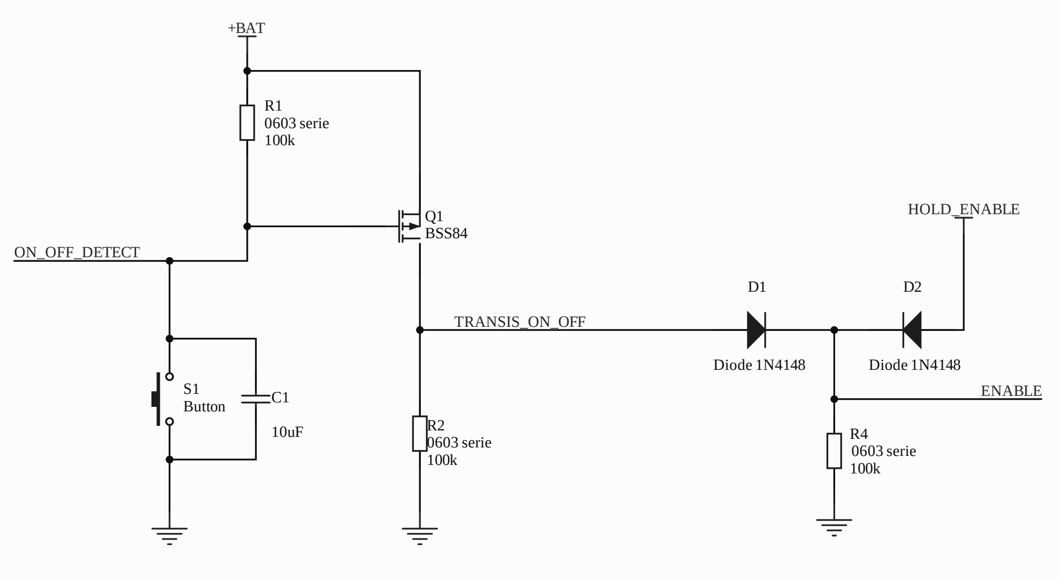


Figure 4.27 – *Power Switch Circuit*

Ideally, what we pretend to do with such circuit is that whenever the button is not pressed, the tension on the BSS84 PNP MOSFET gate is the same as the tension on the source. Consequently, the BSS84 PNP MOSFET would be OFF and the *ENABLE* pin from the AP3429/A module would not have enough tension for supplying the module. In this situations where the buck converter is OFF, all modules will be OFF. When pressing the button for switching ON the device, the gate will be at a GND tension and the source would have the tension of the battery. If this tension is higher than 1.6, the BSS84 PNP MOSFET would be ON and on the *ENABLE* pin from the AP3429/A module would be enough tension for switching ON this module. In this situation, as the buck converter is ON, all modules will be supplied. At this point, as the time passes the battery would discharge itself, therefore we will reach a point in which the battery will not have enough tension for turning ON the BSS84 PNP MOSFET and we will not be able for keeping the AP3429/A module on (consequently, we will not be supplying the modules). For avoiding this situation, we have implemented a microprocessor pin,the HOLD_BATTERY_ON pin, that when the device is turn ON, it is set HIGH. Therefore, on the *ENABLE* pin from the AP3429/A module there will always be enough tension for keeping this module ON and supplying all modules.

Finally, if we set the HOLD_BATTERY_ON pin HIGH, the device will always be ON and we will not be able to shut it off. To dodge this situation, we have implemented a software in which, whenever is detected a GND tension on the gate of the PNP MOSFET after switching ON the device, the HOLD_BATTERY_ON pin will be set LOW.

As this is a circuit we have thought of, we need to test it before implementing it in our design. To do so, we have simulated this circuit in Altium obtaining the following graphics.

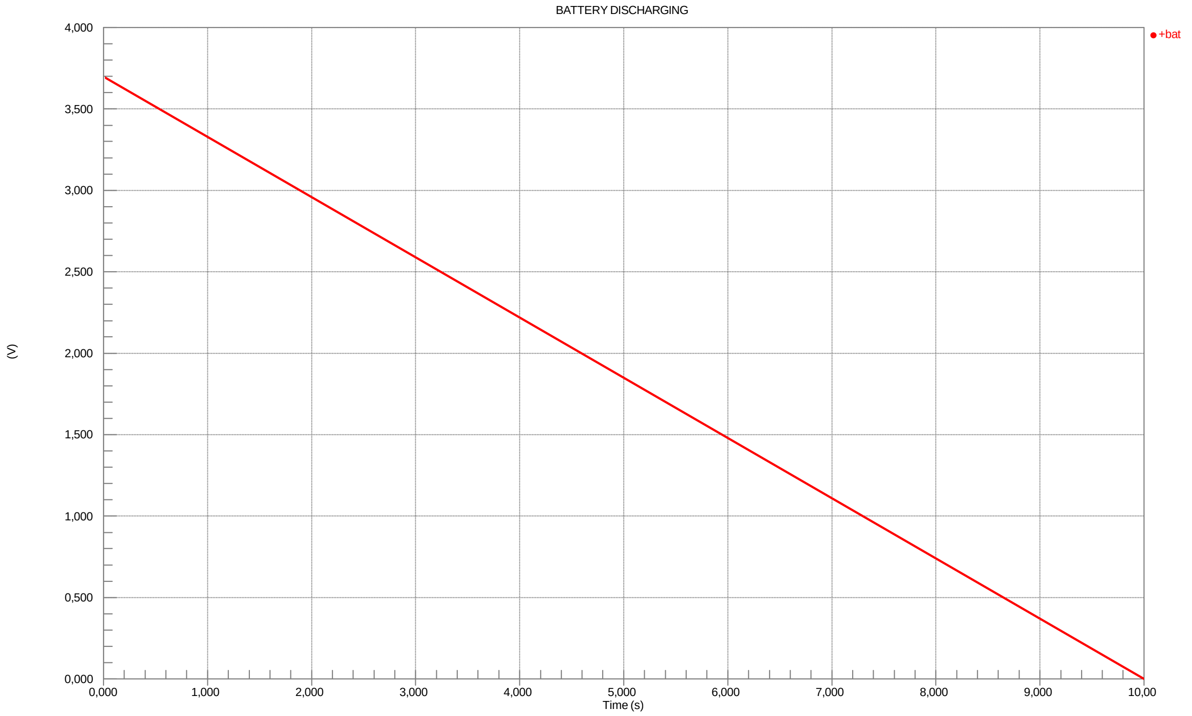


Figure 4.28 – *Battery discharging in the Power Switch Circuit Altium simulation*

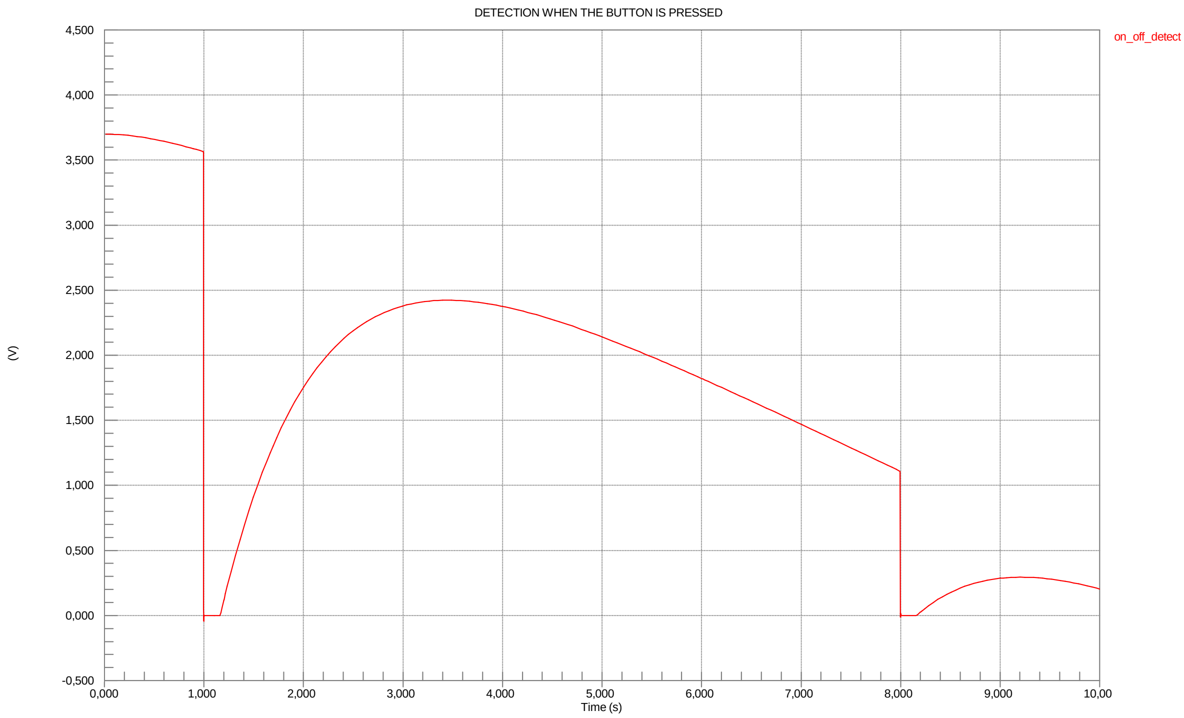


Figure 4.29 – *Button detection in the Power Switch Circuit Altium simulation*

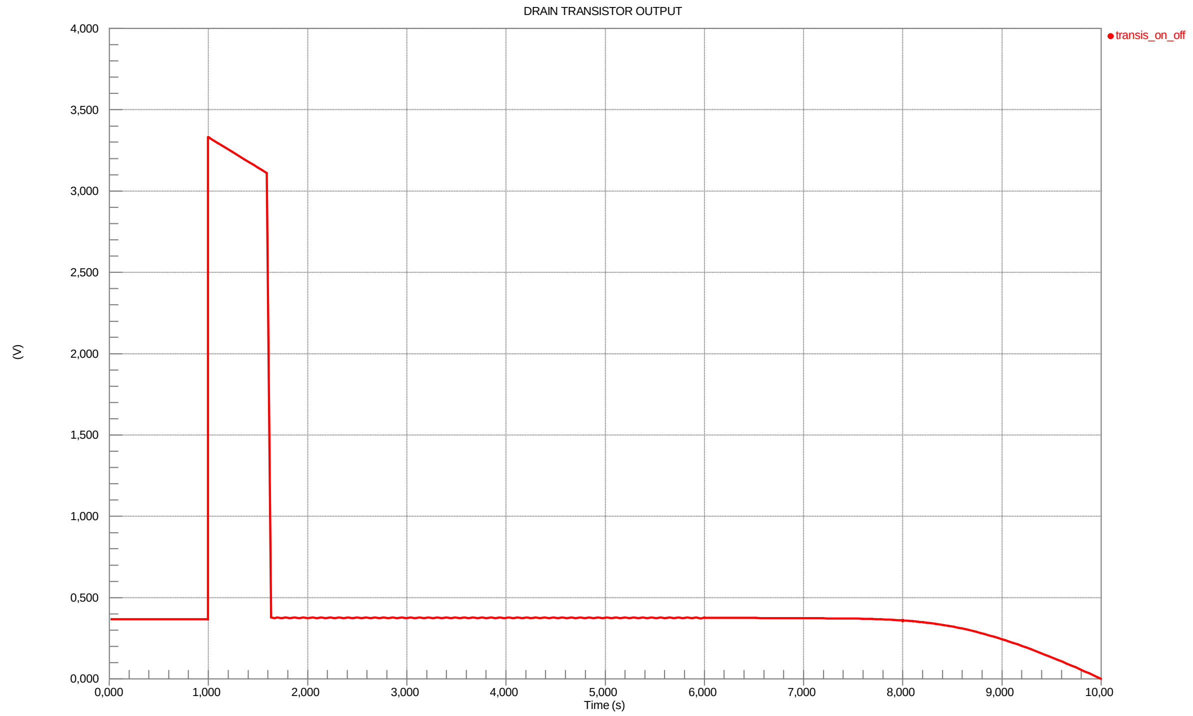


Figure 4.30 – *Transistor behavior in the Power Switch Circuit Altium simulation*

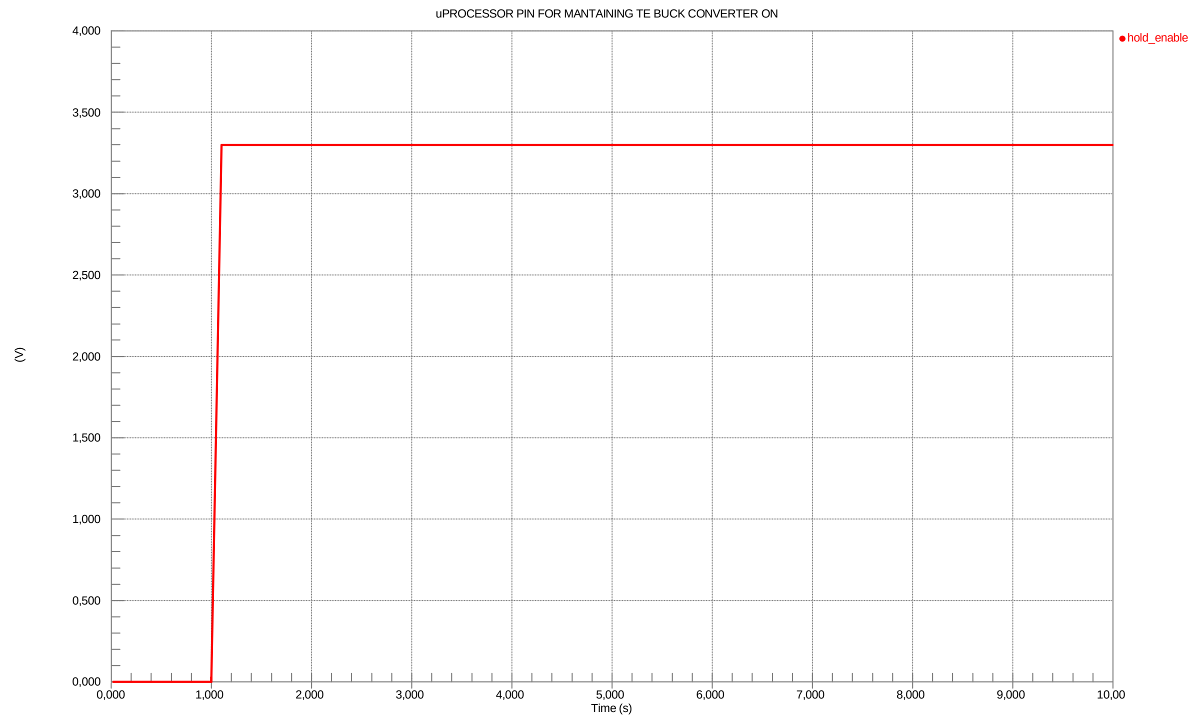


Figure 4.31 – *Hold_Battery_On pin in the Power Switch Circuit Altium simulations*

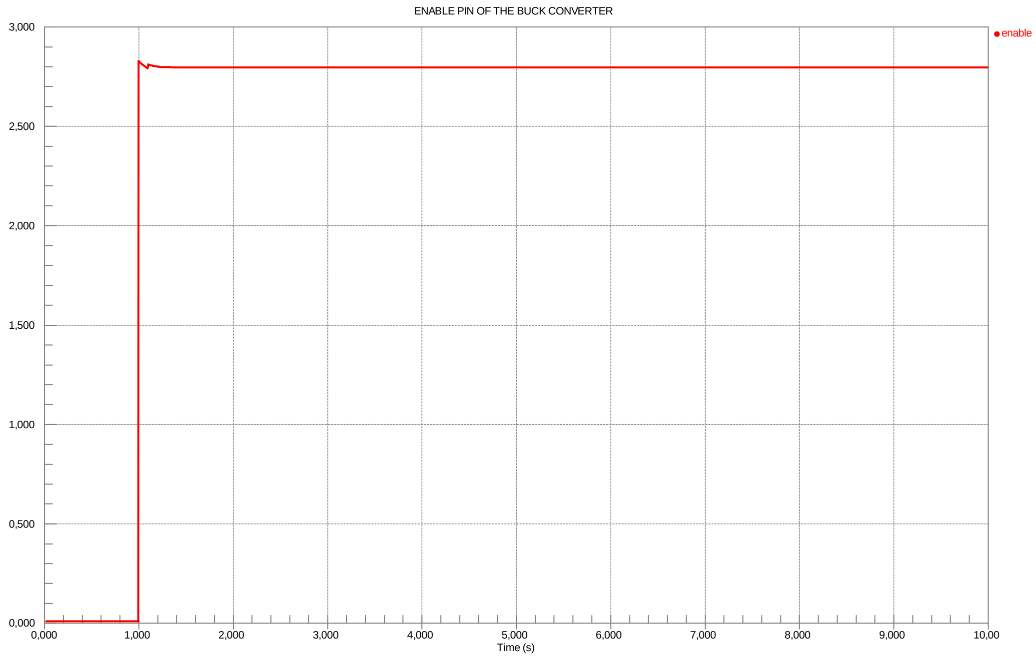


Figure 4.32 – *Enable pin from the AP3429/A module in the Power Switch Circuit Altium simulation*

4

As observed in this images, the behavior is as expected. With the circuit in Figure 4.27 we will be achieving exactly the modules performance we expect.

To close this section, we need to highlight that thanks to the capacitor parallel to the button, we will achieve the behavior we expect from this circuit even when the button is pressed for lees time than the 900 ms that would take for the HOLD_BATTERY_ON pin to be set high.

4.1.1.1.13 Engineering Mode

The last feature we will need to implement in our device is some kind of circuit for entering the engineering mode like all the modern device. This mode will let us change some parameters that the advanced users will consider useful. Entering this mode is as simple as pressing some kind of button combination.

Therefore, for implementing this mode we will only have to add two or more buttons and the rest will relay in the software implementation. We already mentioned in paragraph 4.1.1.1.1 that we were short in pins, so, instead of wasting three pins, we will be implementing a circuit that will let us use two buttons in only one pin. To do so, we will only need an ADC pin, some buttons and some resistors. The circuit is a shown in Figure 4.33.

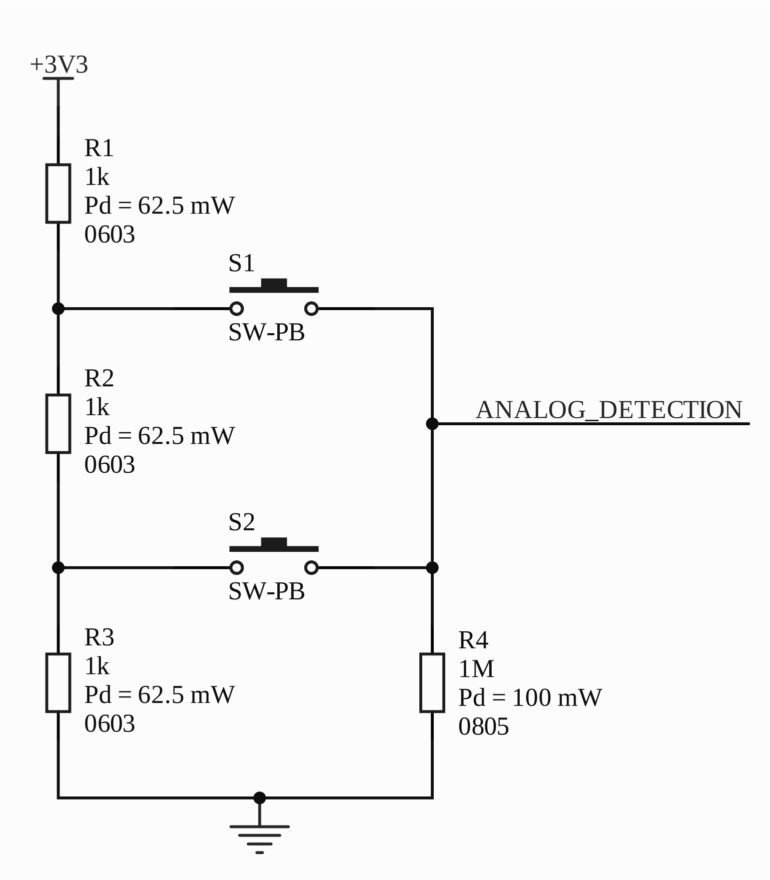


Figure 4.33 – *Circuit for detecting two buttons with only one ADC pin*

As we can observe from this circuit,when no button is pressed, the voltage read on the ADC pin is GND. Depending on which button is pressed, the voltage detected will change therefore, we could program the software to detect the voltage detected depending on which button has been pressed.

When the button we have pressed is S1, the circuit remaining would be the on in Figure 4.34.

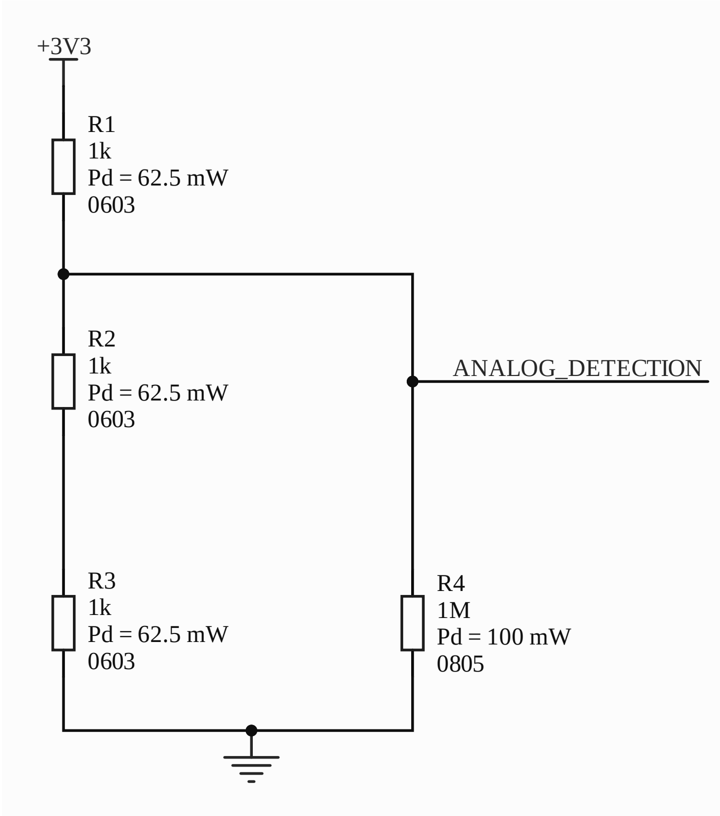


Figure 4.34 – *Circuit implemented whenever the S1 button is pressed*

From this circuit we can solve for knowing which tension will be detected. To do so, we will be simplifying the circuit with the next steps:

$$R_2 + R_3 = 2\ \mathrm{k\Omega}$$

$$(R_2 + R_3) \mid\mid R_4 = \frac{2\ \mathrm{k\Omega} \cdot 1\ \mathrm{M\Omega}}{2\ \mathrm{k\Omega} + 1\ \mathrm{M\Omega}} = 1996\ \Omega$$

$$V_{REF} = 3.3\ \mathrm{V} \cdot \frac{[(R_2 + R_3) \mid\mid R_4]}{[(R_2 + R_3) \mid\mid R_4] + R_1} \quad V_{REF} = 3.3\ \mathrm{V} \cdot \frac{1996\ \Omega}{1996\ \Omega + 1\ \mathrm{k\Omega}} = 2.19\ \mathrm{V}$$

Therefore, the tension detected on the ADC pin when the S1 button is pressed is:

$$\mathbf{V_{REF} = 2.19}\ \mathrm{V}$$

Analogue to what we have done for the case when the button 1 is pressed, we are going to study the tension detected whenever the S2 button is pressed. The circuit that we have whenever this event occurs is the one in Figure 4.35.

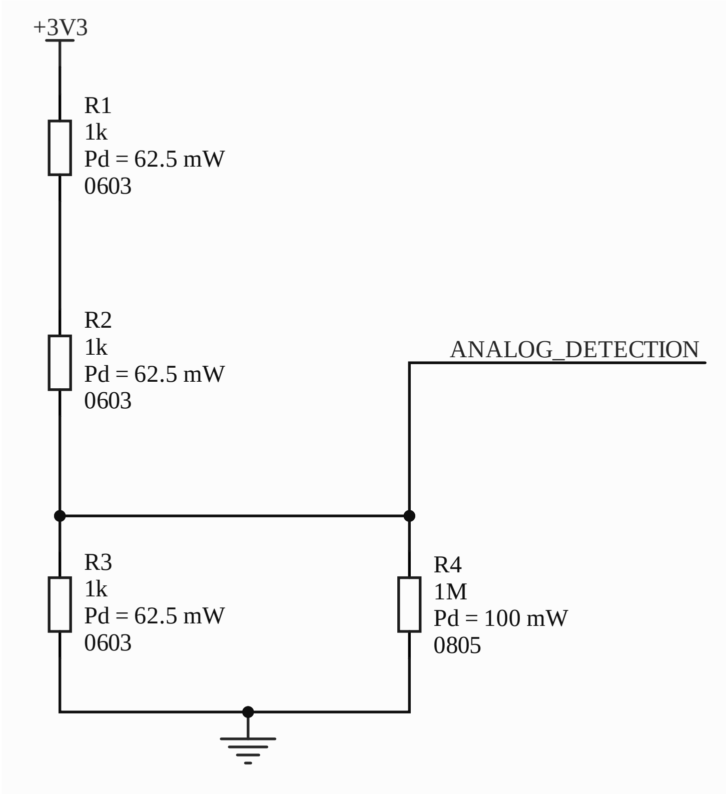


Figure 4.35 – *Circuit implemented whenever the S2 button is pressed*

In this case, the tension detected on the ADC pin can be calculated following the next steps:

$$R_3 || R_4 = \frac{1\ \text{k}\Omega \cdot 1\ \text{M}\Omega}{1\ \text{k}\Omega + 1\ \text{M}\Omega} = 999\ \Omega$$

$$R_1 + R_2 = 2\ \text{k}\Omega$$

$$V_{REF} = 3.3\ \text{V} \cdot \frac{(R_3 || R_4)}{(R_3 || R_4) + (R_1 + R_2)}\ V_{REF} = 3.3\ \text{V} \cdot \frac{999\ \Omega}{999\ \Omega + 2\ \text{k}\Omega} = 1.099\ \text{V}$$

Consequently, the tension detected on the ADC pin whenever the button 2 is pressed is:

$$\mathbf{V_{REF}} = \mathbf{1.099}\ \text{V}$$

The last circuit we could implement is when both buttons are pressed. In this case, the remaining circuit would be the one presented in Figure 4.36.

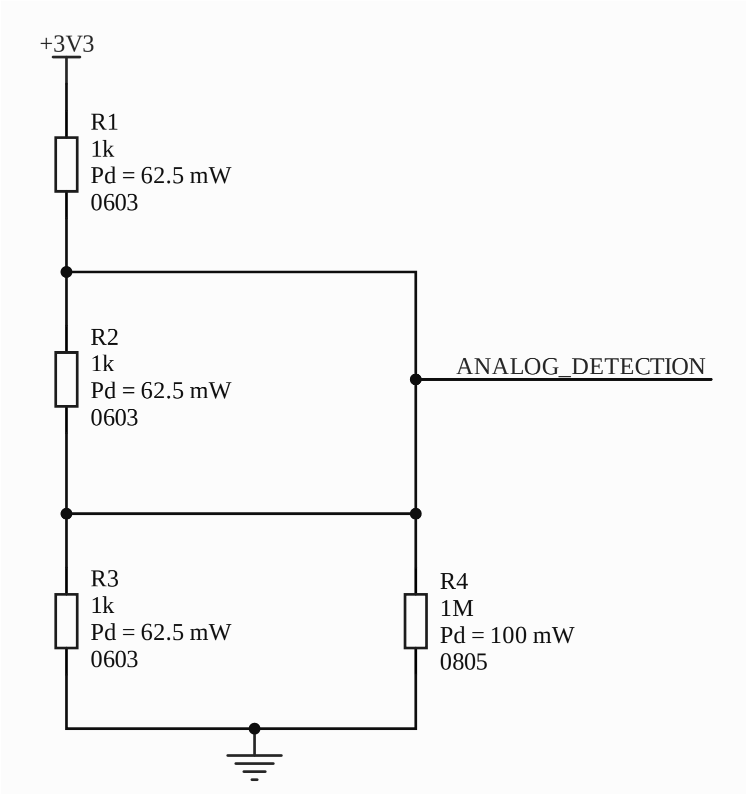


Figure 4.36 – *Circuit implemented whenever the S2 button is pressed*

4

In this case, the steps we have to follow to estimate the voltage detected on the ADC pin are:

$$R_3 || R_4 = \frac{1\ \text{k}\Omega \cdot 1\ \text{M}\Omega}{1\ \text{k}\Omega + 1\ \text{M}\Omega} = 999\ \Omega$$

$$(R_3 || R_4) + R_2 = 1999\ \Omega$$

$$V_{REF} = 3.3\ \text{V} \cdot \frac{[(R_3 || R_4) + R_2]}{[(R_3 || R_4) + R_2] + R_1}\ V_{REF} = 3.3\ \text{V} \cdot \frac{1999\ \Omega}{1999\ \Omega + 1\ \text{k}\Omega} = 2.1996\ \text{V}$$

Consequently, the tension detected on the ADC pin would be:

$$\mathbf{V_{REF} = 2.1996}\ \text{V}$$

With those values we can program the board for detecting which ones are pressed and perfomr a certain event depending on the buttons pressed.

4.1.1.2 Uni Board Hardware

Contrary to the Main Board, this board will hold all extra modules needed for performing the user's secondary mission. Hence, this board will have as much PTH as it can be fitted. Note that he SCH implemented for this board is exhibit in Subsection B.2.3.

As this board is intended for adding extra modules, this board will only lodge two thing: a bus connector for bringing the signals and a visual confirmation (LED).

For connecting more modules, we will be implementing as much 1 mm PTH as possible if we separated by 2.54 mm since this is the standard modules pitch.

4.1.1.2.1 Bus Connector Uni Board

Like we mentioned in paragraph 4.1.1.1.11, we will be bringing some of the signals from the Main Board and Power Board to this board for connecting the new modules to the microprocessor. To do such thing, we will also have to include a bus connector. As we intend to interconnect the three boards, the bus connector for the three boards must be the same, therefore we will stick to the explanation made in paragraph 4.1.1.1.11 to understand this board module.

4.1.1.2.2 LED Uni Board

For checking whether the signals has been brought correctly, we will be implementing a LED circuit. Moreover, this LED could be used by the user in his secondary mission for some purpose.

The circuit implemented in this case is one of the circuits we already studied in paragraph 4.1.1.1.6. This circuit is the one presented in Figure 4.13 just because we could have two different color LED with only one pin.

Just as the circuit implemented is the same, the LEDs used are also the same we used from the laboratory stock (simulated in Section C.1), consequently we will refer to Table 4.8 (BLUE) and Table 4.9 (WHITE) to know more about their features.

4.1.1.3 Power Board Hardware

We have talked about about a bucket converter for turning the battery tension into the 3.3 V tension the modules need, but we have not talked about how are we going to implement the battery circuit or how are we going to charge it.

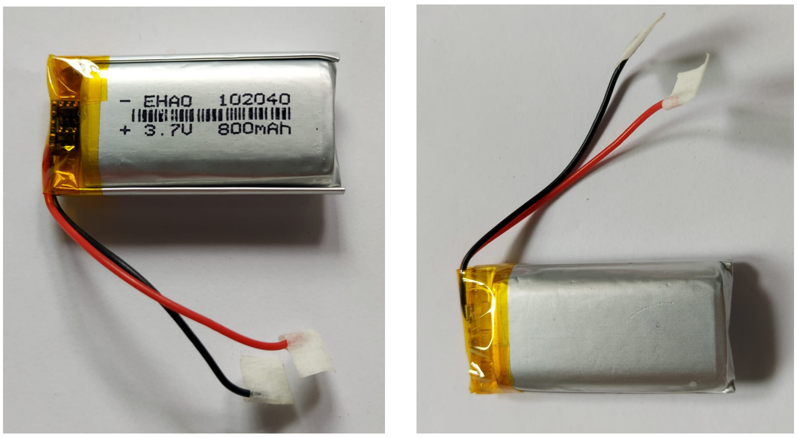
In Subsection 3.2.10 we select the best battery option for our device, a **Flat Li-On Battery**. Also, in Subsection 3.2.11, we mentioned that we are going to be changing the battery charger for a more suitable one.

Before explaining the battery and charger circuit, we need to mention something we already speak of in subsubsection 4.1.1.1. As the ceramic antenna for the GPS we chose (paragraph 4.1.1.1.5) is slightly bigger than the one the Czech engineers used, we place this device on this board instead on the Main Board. However, since our antenna is connected through an SMA connector at the end of a cable, the connection is made on the Main Board. Therefore, on this board we will only be placing the physical device.

Note that the complete SCH files for this board are presented in Subsection B.2.2.

4.1.1.3.1 Battery

The best battery we could integrate in our device mainly for its dimensions is a a **Flat Li-On Battery** (Figure 4.37) like we have mentioned before.



(a) *Front view* **(b)** *Back view*

Figure 4.37 – 404050 *Flat Li-On Battery*

When selecting a battery there are a few key parameters we have to be aware of. This parameters are the one listed below [77]:

- Whether we need a single-use battery (primary) or a rechargeable one (secondary)
- The capacity (mAh) we need for supplying our application the right amount of time
- The nominal voltage our battery should be supplying
- The temperature range for avoiding any risk of explosion

From the consideration we have listed, we should be specially careful with the temperature range since the Lithium batteries are very inflammable one and if the very thin plastic wall that separates the positive cathode of the negative anode suffers any damage, the cathode and anode will touch each other and explode.

The main characteristics of the battery are the one exhibited in Table 4.19.

	102040 Flat Li-On Battery
NOMINAL VOLTAGE	3.7 V
MAXIMUM VOLTAGE	Up to 4.25 V
CAPACITY	800 mAh
TEMPERATURE RANGE	– 20 to 60 °C
CHARGE METHOD	STANDARD : 0.5C FAST: 1.0 C
MAXIMUM CHARGING CURRENT	1.0 C
MAXIMUM DISCHARGING CURRENT	2.75 V
DISCHARGE CUT-OFF VOLTAGE	1.0 C
CHARGING TIME	STANDARD : 3 to 7 hours FAST: 3 hours
PROTECTING FUNCTION	overcharge, overdischarge, overcurrent, short circuit protection, overheat protection
PROTECTING CIRCUIT	Integrated PCM for avoiding the overcharge or overdischarge
DIMENSION	10 x 20 x 40 mm^3 15 g
PRICE	5.10 € https://es.aliexpress.com/

Table 4.19 – 102040 ***Flat Li-On Battery*** *main characteristics [34]*

The main things we need to highlight from Table 4.19 are the capacity, the tension and the protecting circuit. As our module's device are powered-up with 3.3 V we do not need a high voltage battery. Therefore, the 3.7 V battery is the perfect tension for us.

The 800 mAh capacity has been selected taken into account the power budget (Subsection 4.1.3). This capacity will allow us to have an autonomy of 1 hour and 8 minutes since our typical consumption is 700 mA. If all modules were working at its maximum consumption, the autonomy in this case would be 24 min. Moreover, if the user introduce any more modules, the autonomy would decrease as the consumption will raise.

The last feature of the Table 4.19 that we would like to highlight is protection circuit module (PCM) that the battery integrate. This circuit can be seen if we look a little closer into the battery's yellow part. This circuit is intended for protecting the circuit and the battery for overcharging or overdischarging itself. As it is integrated on the battery, we would be saving the corresponding space on the PCB.

4.1.1.3.2 Battery Charger

The last module we will be commenting from the CanSat transmitter is the battery charger. As we already introduced in Subsection 3.2.11, the module we will be using is the **TP4056** (Figure 4.38). This device does not only integrate the **TP4056** charger chip and the corresponding passive elements, but also two extra chips for protecting the battery form a possible short circuit and also it integrates overcurrent protection. The chip needed to achieve this protection is the **DW01A** chip. The other chip integrated in the **FS8205A** which is required for the DW01A chip.

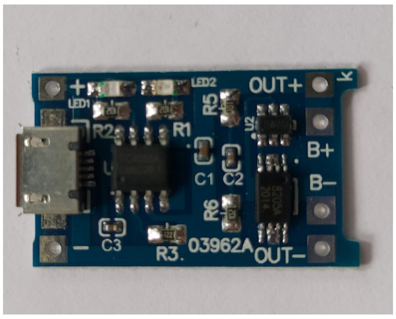


Figure 4.38 – ***TP4056*** *charger module*

This module showed in Figure 4.38 integrate his own USB but as we only want to have one USB connector in our design, we will be desoldering its components and placing them in our Power Board PCB. At this point we may wonder why not buy the components separately instead of buying the whole module. The reason for buying the whole module is just because it is cheaper buying the whole module than buying the chips individually. We will be buying it from `es.aliexpress.com` and the price for the whole module is 0.314 €.

Once this has been said, we will be explaining the circuit implemented. As this module is a circuit formed by the interconnection of the three modules sub-circuits, we will be dividing it into each module circuit even though the complete circuit is presented in Subsection B.2.2. Although before going on to comment the different circuits, we must said that for better understanding this circuit, we have extracted all his information from the `best-microcontroller-projects.com` site where they have dedicated a whole entry for explaining this module [58].

4.1.1.3.2.1 TP4056

This chip is the main device needed for charging the battery. The **TP4056** is a complete constant-current/constant-voltage linear charger for lithium-ion batteries [57]. The main characteristics we need to take into account when using or even choosing this module are summarized in Table 4.20.

	TP4056
SUPPLY VOLTAGE (INPUT)	4.2 to 8.0 V
SUPPLY CURRENT	Charge Mode: 500 µA Standby Mode: 100 µA Shutdown Mode: 100 µA
REGULATED OUTPUT VOLTAGE	4.2 V (4.264 V max)
TEMPERATURE RANGE	− 40 to 85 ºC
LED CHARGE STATE	RED LED –> Charging GREEN LED –> Charged
DIMENSION	5.5 x 6 x 1.5 mm^3 (SOP-8) 77.4 mg

Table 4.20 – ***TP4056*** *main features [57]*

As for the circuit implemented is the one suggested by the datasheet. This circuit is the one showed in Figure 4.39.

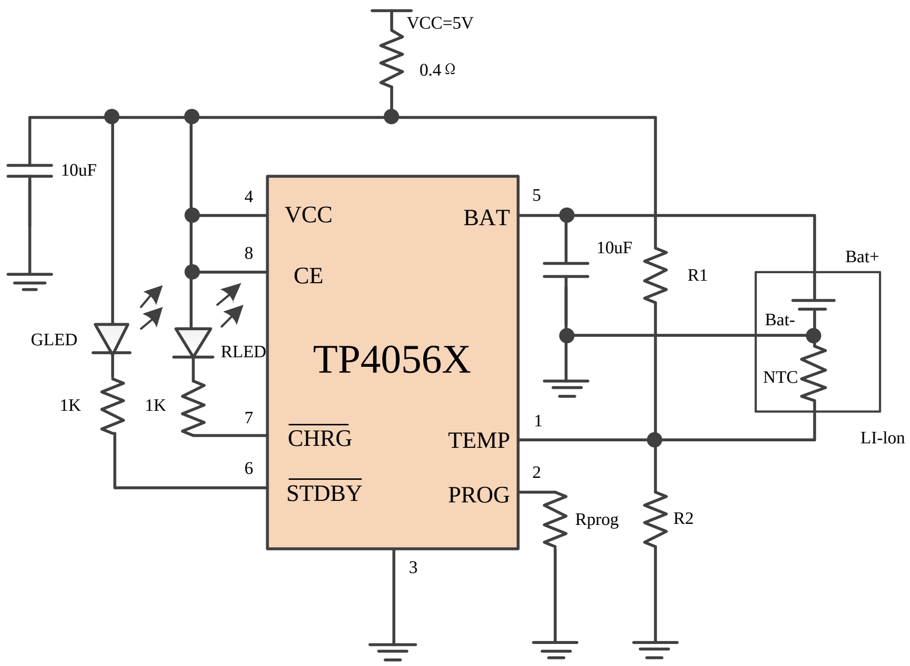


Figure 4.39 – ***TP4056*** *suggested circuit [57]*

When explaining the reasons for choosing this module in Subsection 3.2.11 we mentioned that we could change one resistor for changing the current. The resistor we must modify in this case is the R_{prog} and the values we could achieve depending on this resistor values are according to the datasheet the ones presented in Table 4.21.

R_{prog} (kΩ)	I_{BAT} (mA)
10	130
5	250
4	300
3	400
2	580
1.66	690
1.5	780
1.33	900
1.2	1000

Table 4.21 – ***TP4056*** *R_{prog} values* [57]

4

As we want the maximum current, we will be setting this value with R_{prog} = 1.2 kΩ.

The last thing we need to mention about the circuit in Figure 4.39 is that as we are going to add two extra modules for protection, we will omit the R1 and R2 resistors since the datasheet only recommends them (they are not crucial).

4.1.1.3.2.2 DW01A

This chip is included for protecting the module from short circuit and overcurrent. Moreover, this module is able to provide the circuit two main features: charger input protection and battery monitoring.

The charge input protection this module accomplish consists of:

- Short Circuit detector
- Over Current detector
- Charger detector
- Reverse charger detection

On the other side, the battery monitoring function this chip provide is related to:

- Overcharge detector

- Overdischarge detector

The main characteristics we may highlight from this module are the one listed in Table 4.22.

	DW01A
DRAIN-SOUCE VOLTAGE	20 V
GATE-SOURCE VOLTAGE	± 10 V
DRAIN-SORUCE CURRENT	6 A
POWER DISIPATION	1.5 W (max)
DIMENSION	6.4 x 3.1 x 1.2 mm^3 (TSOP-8) 33.4 mg

Table 4.22 – ***DW01A** main characteristics [37]*

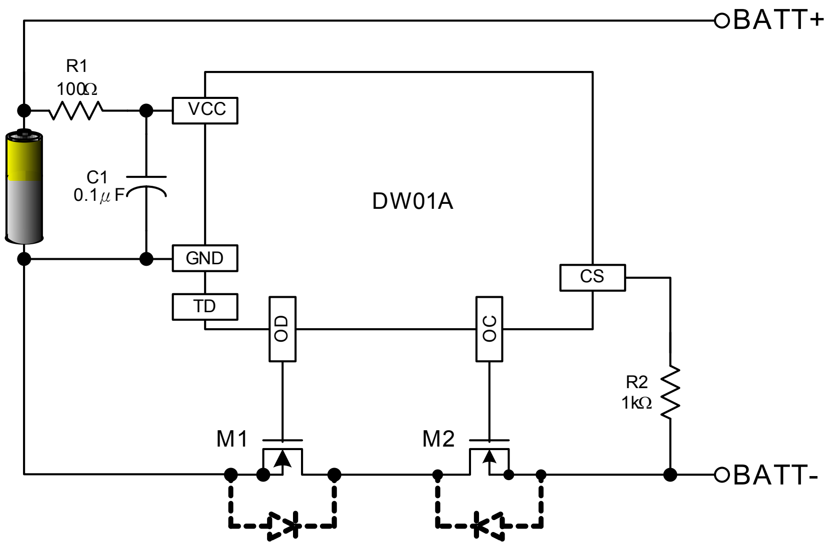


Figure 4.40 – ***DW01A** application circuit [37]*

The only thing we need to highlight from this module is that, the two required MOSFET (M1 and M2), are being introduced with the FS8205A module (subparagraph 4.1.1.3.2.3) instead of using two separate elements.

4.1.1.3.2.3 FS8205A

This chip is a dual N Channel MOSFET required by the DW01A chip. Therefore, the connection is pretty simple and is already showed in Figure 4.40.

The most relevant characteristics of this module are the ones exhibit in Table 4.23.

	FS8205A
OVERCHARGE PROTECTION VOLTAGE	4.25 to 4.35 V
OVERCHARGE DELAY TIME	200 ms
CURRENT	Supply Current: 6.0 µA Power-Down Current: 4 µA
TEMPERATURE RANGE	– 40 to 85 °C
DIMENSION	2.5 x 3.0 x 1.1 mm^3 (SOT-23) 8 mg

Table 4.23 – ***FS8205A*** *main features [39]*

4

4.1.1.4 Base Board Hardware

Now we have finished introducing all the CanSat transmitter modules, we are going to do the same process with the CanSat receiver. The board that will act as the CanSat receiver is the **Base Board**.

The Base Board will hold all the required modules for the receiver to work. Since on this board we do not have any space restriction (does not need to be introduced into a "soda" can), we will be lodging this board all modules including the power supply circuit.

The SCH of this board are presented in Subsection B.2.4.

Some of the modules we introduced in the Main Board are the same as the one we will be implementing on this board, therefore for those modules we will stick to the explanation we did on the corresponding Main Board section. The only module that we will not be implementing on this board is the temperature, pressure, humidity and altitude measuring sensor. Instead, we will be introducing a new module, a LCD for presenting the values measured and received via radio.

Without any further ado, we are going to start the circuit's explanation. To do so, we will be explaining them by the order of appearance in the SCH as we did for the other boards.

4.1.1.4.1 Microprocessor

For this board we will be using the **ESP-32-WROOM-32D** microprocessor as we did on the transmitter. For more information about the main characteristics or the circuit we will be referring to paragraph 4.1.1.1.1.

The only thing we have changed from the transmitter is the pin assignment which is exhibit in Subsection B.2.4. Moreover, as the LCD module requires more pins than the BME280 sensor we remove, we will be needing some more pins. Since we already have demonstrated in paragraph 4.1.1.1.1 that the pins used for the SPI Flash, which are not recommended for other uses, can be theoretically used, we will be taking specially attention with those pins assignment.

4.1.1.4.2 MicroSD Card Socket

Even though we may think that this module is meaningless in this board since we observe the data received or measured through the serial monitor on the PC, this is a mistake. Of course we are in touch with the data received or measured, but we need to save them for further analysis. The microSD will allow us to store the strings received via radio and to compare them with the data saved in the transmitter's microSD to see how much information we have missed.

As for the information regarding the main features of the **MOLEX 504528** module or even the circuit implemented we will be remitting to paragraph 4.1.1.1.3.

4.1.1.4.3 Radio Receiver Module

One of the most relevant modules of the receiver board is the radio module since it is the one in charged of receiving the measured data with the transmitter board. To do so, analogue to the transmitter radio module, we will be using the **RFM69W**. Therefore, the most relevant features of this module or even the circuit can be consulted in paragraph 4.1.1.1.4.

The only thing that will differentiate the transmitter from the receiver is the software implemented. The hardware will be the same given that for establishing the communication with both modules we must have the same frequency and channel. Otherwise we will not get the boards to communicate with each others.

To conclude with this module, we must remark that even though on the transmitter we implemented an antenna made only by a simple electronic wire (with the right length), on this module we will be adding a **433 MHz SMA antenna ??**.

(a) *Front view* **(b)** *Back View*

Figure 4.41 – 433 ***MHz SMA antenna*** *module [30]*

4

The main features we will be highlighting from this module are:

	433 MHz SMA antenna
GAIN	3.0 dBi
INTERFACE	male SMA convertible to IPX
FREQUENCY	433 MHz
RESISTANCE	50 Ω
DIMENSIONS	105 x 13 mm^2 14.8 g
PRICE	2.67 € https://es.aliexpress.com/

Table 4.24 – 433 ***MHz SMA antenna*** *main features [30]*

For connecting the antenna to the board, we will be placing an SMA female connector. This connector is the one presented in Figure 4.42.

(a) *Front View*

(b) *Back View*

Figure 4.42 – *SMA antenna connector*

Note that in our kit we will be including this antenna contrary to what the Czech engineers did. They did not include any antenna for the receiver radio, therefore the boards could only be tested if the user buy an antenna. This was inconvenient since users could not get into programming until they had built the antenna, which took away learning time.

4

4.1.1.4.4 GPS Receiver Module

A great idea for drawing a map with the CanSat transmitter trajectory is to implement a GPS module for fixing the receiver coordinates. The difference between the receiver fixed coordinates (we will not move this board) and the transmitter variable coordinates will allow us to draw this trajectory.

For getting the coordinates of the receiver we will be using the same module as the transmitter (implementing the same circuit). Thus, we will hand on its explanation to paragraph 4.1.1.1.5.

Analogue to the radio module, the Czech engineers did not include the antenna, only the connector. In our design we will be including both the SMA connector (Figure 4.42) and the antenna. The most economic antenna we found for our device is the **FAKRA-C active antenna** (Figure 4.43).

(a) *Top View*

(b) *Back View*

(c) *Connector View*

Figure 4.43 – ***FAKRA-C active antenna*** *module [40]*

The main features we have consider when selecting it are listed in Table 4.25.

	FAKRA-C active antenna
POWER SUPPLY	3.0 V to 5.0 V
CURRENT CONSUMPTION	10 mA
FREQUENCY	1575/1561 MHz
GAIN	38 dBi
NOISE	1.5 dB
TEMPERATURE RANGE	– 40 to 85 °C
CABLE	3.0 m
INTERFACE	SMA male connector
DIMENSIONS	105 x 13 mm^2 14.8 g
PRICE	5.29 € https://es.aliexpress.com/

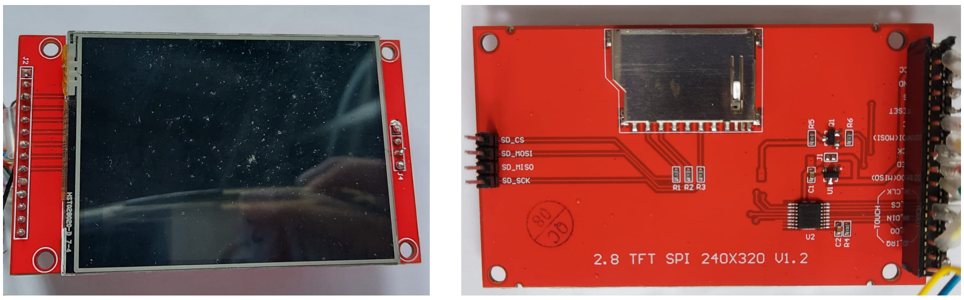
Table 4.25 – ***FAKRA-C active antenna*** *main features [40]*

4.1.1.4.5 TFT LCD + SD

The only module we will be introducing in this board that differs from the ones used in the transmitter is the display module. This module has been introduced as a way of presenting nicely the information received by the radio module. Also, we can display the information about the GPS coordinates or even the current consumption measured with the INA219 module.

This module was also introduced by the Czech engineers but the one they choose is smaller than the one we have chosen as we already discuss in Subsection 3.2.6.

The display module we have chosen is the **ILI9341** TFT LCD + SD module (Figure 4.44). That means that we will have not only a LCD but also we will have an extra SD module in case the images displayed in the LCD are too heavy and fill up the microSD card intended for storing the received data.



(a) *Front view* **(b)** *Back view*

Figure 4.44 – ***ILI9341*** *TFT LCD + SD module [42]*

The main characteristics we will be highlighting from this module are listed in Table 4.26.

	ILI9341 TFT LCD + SD
POWER SUPPLY	2.7 to 3.3 V
PFORWARD CURRENT	80 mA
TEMPERATURE RANGE	– 20 to 70 ºC
LCD SIZE	2.8″
RESOLUTION	240 x 160
DISPLAY COLOR	RGB 65K color
TOUCH SENSOR	Yes (XPT2045 Controller)
INTERFACE	SPI / I2C

Continued on next page

	ILI9341 TFT LCD + SD
DIMENSIONS	50.2 x 86 mm^2 36 g
PRICE	12.75 € https://es.aliexpress.com/

Table 4.26 – *ILI9341 TFT LCD + SD main features [32]*

The main features we want to remark is that this module can have both interfaces I2C or SPI. Since we are short in microprocessor pins, we will be interfacing it through the I2C interface since this does not require an extra pin (the CS pin) even though it is a slower communication. Another thing we want to highlight is that this module comes with a touch sensor which will save us for the buttons space and budget.

For implementing this module circuit there are three thing we must consider: the LCD interface, the Touch Sensor interface and the SD interface. Those three features shall be connected separately.

For interfacing this module with the microprocessor we must follow the connecting listed in Table 4.27.

ILI9341 TFT LCD + SD Pins	ESP-32-WROOM-32D Pins
VCC	+ 3.3 V power supply
GND	GND
CS	GPIO39
RESET	GPIO32
D/C	GPIO35
SDI(**??**)	I2C SDA
SCLK	I2C SCL
LED	+ 3.3 V power supply
SDOK(MISO)	Not Connected
T_CLK	SPI SCLK (HSPI)
T_CS	GPIO26
T_DIN	SPI MISO (HSPI))
T_DO	SPI MOSI (HSPI)
T_IRQ	Not Connected
SD_SCLK	SPI SCLK (HSPI)
SD_MISO	SPI MISO (HSPI)
SD_MOSI	SPI MOSI (HSPI)
SD_CS	GPIO27

Table 4.27 – *ILI9341 TFT LCD + SD Module interface to* ***ESP-32-WROOM-32D***

To conclude, we want to remark that the SD and the Touch Sensor are interfaced via the HSPI whereas the LCD is interfaced via I2C protocol.

4.1.1.4.6 LEDs

It is important in any electronic device to add some visual indicator for certain events. The visual indicator we are going to use is some LEDs circuit for indicating some relevant situations.

The LEDs we are going to be using are the ones available on the laboratory stock. Moreover, the circuit we will be implementing for indicating certain events are the one we implemented in the transmitter. Hence we will be referring to paragraph 4.1.1.1.6 for learning more about this events or the circuits implemented.

4.1.1.4.7 Buzzer

On the receiver we will also add a buzzer. However, we will not be using it for recovery purposes as this board should not be taking any risk of being lost. The use we will be giving to this module on the receiver board is only for indicative purposes.

Some of this indicative purposes we could give this board is, for example, making a sound when the transmitter starts the communication or indicating when the battery is low.

Analogue to the other modules, we will be using the same component, the **KXG1205**, as the transmitter. Consequently, the information about the characteristics or the circuit is in paragraph 4.1.1.1.7.

4.1.1.4.8 Current \Power Monitor

Like we are using the same modules in the transmitter and receiver, it could be a good educational exercise to be able to compare the consumption of both boards measured with the same component, the **INA219**. To do such thing, is important to save both the transmitter and receiver INA219 measurements in a microSD file.

For further information about the module or the circuit, we will remit to paragraph 4.1.1.1.8.

4.1.1.4.9 Hardware Reset

On this board we have also added a reset button and a boot button in case we need it. Analogue to the ones on the transmitter board, we have performed the same reverse engineering. Therefore, the circuits implemented for both buttons are the same since the microprocessor is also identical.

To acknowledge more about the circuit implemented for both buttons we will recommend to read paragraph 4.1.1.1.9.

4.1.1.4.10 USB + Power Management

Different from the CanSat transmitter, we will be implementing the power system on the same board as the rest of the modules. Thus this section will include not only the USB to serial conversion, for communicating the board with the PC,and the circuit needed for converting the 3.7 V battery tension to the 3.3 V supply tension, but also the battery and charger modules.

Although we will be implementing the power circuit in this board, all the modules included in this section are the same used in the transmitter. Consequently, the circuits are also the same. Hence we will refer to their respective section for learning more about them.

4

- Information about the **MOLEX47346 − 0001** in subparagraph 4.1.1.1.10.1
- Explanation of the USB to serial conversion with the **CH340** chip in subparagraph 4.1.1.1.10.2
- For the programming circuit we will direct the reader to subparagraph 4.1.1.1.10.3
- The **AP3429/A** characteristics and implemented circuit are in subparagraph 4.1.1.1.10.4
- The battery explanation can be found in paragraph 4.1.1.3.1
- The **TP4056** battery charger information is in paragraph 4.1.1.3.2

4.1.1.4.11 Bus Connector Base Board

In case the user wants to add some extra modules for which there are no space, we are going to implement a bus connector. This way, with some the user will be able to add any module he wants. As this board is on the ground, preferably static, we would not be taking any risk of disconnecting this wires.

Analogue to the CanSat transmitter, we will be sending the same signals so that the user can add any type of module. Therefore, for consulting the information about the pin header size or the signals we will be implementing for extra modules, we will refer to paragraph 4.1.1.1.11.

4.1.1.4.12 Power Switch

Following the same argumentative line in which we have been using the same modules, and therefore the same circuits, we need to find some sort of system to get to turn ON or OFF the device whenever we want. The circuit implemented for such purpose on the CanSat transmitter is the most effective method we could think of, therefore we will remit to paragraph 4.1.1.1.12 for learning more about this circuit.

4.1.1.4.13 Engineering Mode

As we discuss in previous sections, nowadays, no electronic device can be conceived without a button combination for entering a menu that allows the user to change the system settings. Since we are designing an updated device, we will be implementing this feature.

The system implemented for accessing the engineering mode has already been explained in paragraph 4.1.1.1.13.

4.1.2 PCB Design

This section will include the process for designing the PCBs. This process consists of two parts: circuit designing and modules placement. For both process we will be using Altium.

The circuit design consists of implementing the circuit needed for the modules to work on SCH files. This is the process we have described in Subsection 4.1.1.

Once the circuits have been implemented, we need to place them on our PCB. This is not a banal process since we have to distribute the available space to have the most functional and efficient design. This is the process we are going to be describing in this section.

Using Altium as the tool for designing any electronic device allow us to automatically link the .SCH files to the .PCB files. Therefore, once we have made modules circuits' on the SCH files, we can start placing this circuit on the board for routing them.

On Altium this connection between the SCH files and the PCB files is made by linking the footprints to the SCH symbols. We must remember that the footprint is the arrangement of pads or through holes used to physically attach and electrically connect a component to a PCB. The land patterns on a circuit board matches the arrangement of leads on a component. The way Altium associates the SCH symbols to the footprint is via pin numbers and pad numbers respectively. Consequently, is especially important to maintain the pin number notation on both the SCH symbols and on the footprint.

Furthermore, as the footprint is the physical representation of the element, the size must be similar to the real element. For implementing each module footprint we should follow the datasheet recommended PCB land pattern.

As an example of all this process, we will refer to Figure 4.45 for the SCH symbol of the microprocessor. The footprint associated to this symbol can be observed in Figure 4.46. Note that for implementing the footprint we have followed the PCB land pattern on the datasheet (Figure 4.48) [65]. The last thing we have to mention it that Altium allow us to observe the 3D model of our design, therefore for each module we need to also inlcude the 3D model (Figure 4.47)

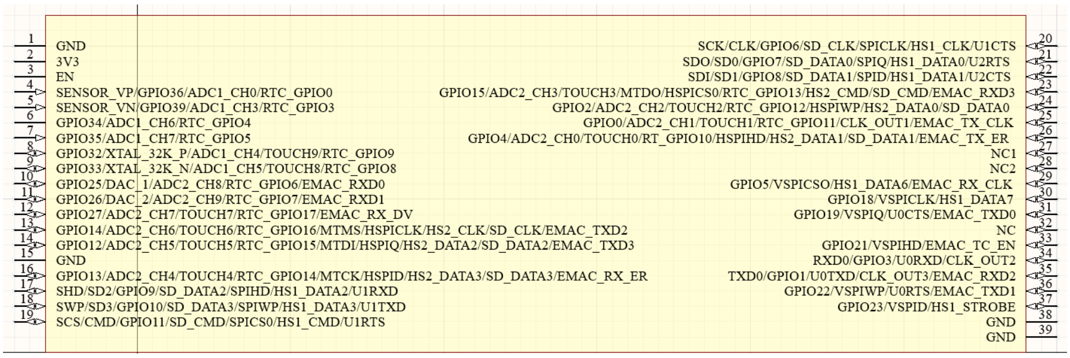
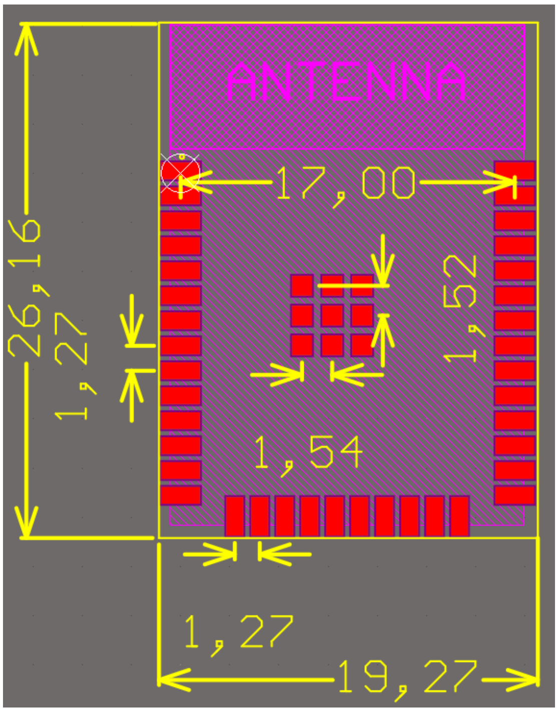
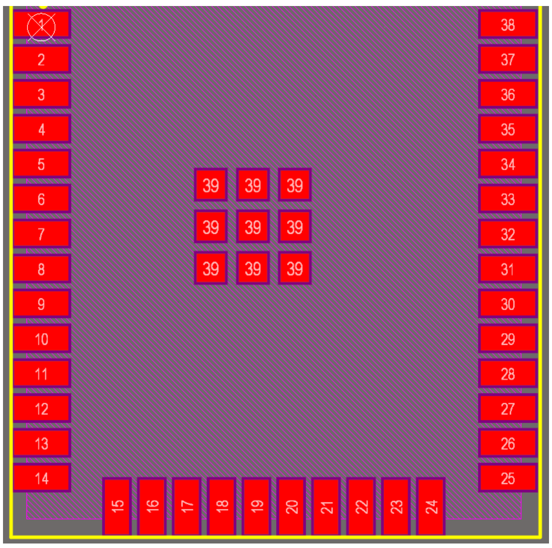


Figure 4.45 – *ESP-32-WROOM-32D SCH symbol*



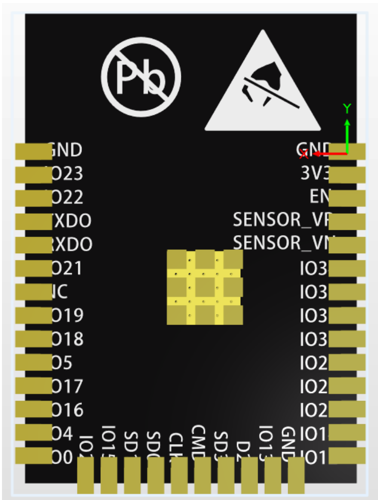
(a) *footprint with dimensions*



(b) *footprint pads numbers*

Figure 4.46 – *ESP-32-WROOM-32D footprint*

(a) *Front*



(b) *Back*

Figure 4.47 – *ESP-32-WROOM-32D 3D footprint view [38]*

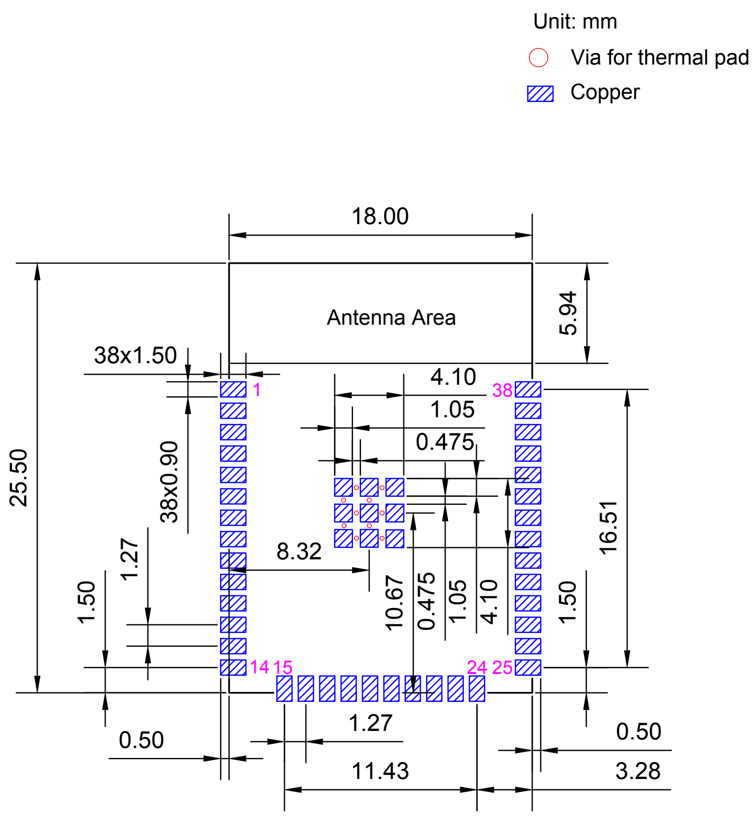


Figure 4.48 – *ESP-32-WROOM-32D recommended PCB land pattern [65]*

Once we know how Altium implements the link between the SCH and the PCB, we can move on to highlight the main things we need to take into account when designing our PCB. This considerations are going to be listed in subsubsection 4.1.2.1.

4.1.2.1 Steps for a good PCB design

Like we have mentioned, whenever we are making a PCB we need to consider the points listed below:

- **PCB production Technology:** The main item we need to consider before starting to design our PCB design is the technology we are going to be using to do such a thing. In our case we can choose between manufacturing in the laboratory or having it produced in China. Since the elements we are going to use are too small, we will proceed to send it to be produced in China. This decision has been made taking into account the fact that in the laboratory we do not have the possibility of finishing the PCB with a solder mask layer, therefore the space between elements for manual soldering should be large enough not to risk making any short circuit. As the space is something that in our design does matter since we have a requirement that imposes the maximum size of our PCB, we can not have the

4

luxury of separating thing too much, consequently it is necessary to have this final layer. To sum up, our production is going to be made by *jlcpcb.com* [46] manufacturer.

- **Capabilities:** Once we have chosen the design technology, we need to know the features of the PCB. This capabilities depends on the production technology. In our case, the capabilities we have chosen are exposed in Table 4.28. As we have sent it to a manufacturer, we had the possibility to choose some of this features such as the PCB thickness. If we had manufactured it in the laboratory we could not have chosen many of these characteristics since the machine does not have this possibility.

Layer Count	2
PCB Qty	5
Material	FR-4 Standard Tg 130-140C
Dielectric Constant	4.5 (double-side PCB)
Dimension	rounded : 60 x 60 rectangular : 100 x 70
Board Thickness	1.6 mm
Thickness Tolerance	± 10 % ± 0.1 mm
Finished Outer Layer Copper	1 oz (35 µm)
Finished Inner Layer Copper	0.5 oz (17.5 µm)
PCB Color	Green
Silkscreen	White

Table 4.28 – *PCB capabilities chosen in jlcpcb.com [46] manufacturer*

- **PCB design rules:** Another item that depends on the production technology and that we have to take into account whenever designing a PCB is the designing rules. This rules will give us the space needed between each element, the maximum an minimum hole size, the minimum and maximum track size, etc. In our case, this rules can be directly downloaded from their site and imported to Altium. The rules can be found in `jlcpcb.com/Capabilities` [46]. Some of this rules are as shown in Figure 4.49.

```
 Design rules
 ------------
- 2 layer
        - 2 oz copper
        - 8 mil trace with & clearance
        - 0.3 mm min. hole diameter
        - 0.6 mm min. via diameter
        - 0.25 mm hole clearance

Layer stackups
 ------------
- 2 layer
        - 2 oz copper
        - 1.6 mm, 2.0 mm min. hole diameter
```

Figure 4.49 – *jlcpcb.com manufacturer designing rules [46]*

- **Mounting Technology:** Once we know the PCB production features and their designing rules, we will need to decide which mounting technology is best for our device. Since the space and weight is an issue we need to consider, the best technology is using SMD components. Even though this is the best technology, some of the elements are not available on this techonology (such as the buzzer) or are required to be THT (such as the pin header for connecting the different boards), therefore we will be using also THT although in less proportion.
- **Package types:** Depending on the device we are going to use, different packages may be available. Therefore we will need to be choosing which one we want to use. For instance, the SMD resistor and capacitor can be found in different sizes such as 1005, 0201, 0402, 0603, 0805, 1008, etc. For choosing which is best we will be considering whether we plan to be soldering them manually or not. As we intend to solder them manually we will choose the smallest size we could solder easily by hand (usually we will use 0603 capacitors and resistors which are the smallest size available on the laboratory stock).

Once all of this have been selected, we can move on to the PCB modules' placing and routing.

4.1.2.2 PCB module's placing

Even though we may think this is a banal process that only requires to fit all modules into the boards, is not so. This is a critical process when designing a PCB that will set the boards efficiency and comfort.

When thinking how to place the modules on the board,the main thing we need to consider is how to place them for facilitating the routing. Furthermore, for creating a comfortable and efficient design, we will need to consider that some modules (such as the USB connector) requires to be place in a determine position to be easy to use.

Even though we may think that the first thing (facilitating the routing) is the most important aspect to consider, some modules requires to take precedence its position on the board for a comfortable and efficient design. This modules and their requirements are listed below:

- **USB connector:** For a good USB connection we need to consider to place this connector somewhere on the board edge of the top layer. Moreover, if we want to keep the USB cable from interfering with the board manipulation, we should consider placing it somewhere on the back, behind from where we would place our hands.
- **Bus Connector:** This module is though for connecting different boards. Since this module is a bit taller, we need to consider placing it somewhere in the back, where this module would not disturb the buttons manipulation.
- **Buzzer:** Analogue to the bus connector, this module is tall enough for disturbing the buttons manipulation, therefore we will think of placing it far from the buttons.
- **LEDs:** As this are used as visual indicators, this modules should be placed on the top layer, at the front of the board.
- **GPS Antennas:** When choosing where to put the GPS antenna we must think on placing them on the top layer as they require to be place facing the sky. Furthermore, if we do not want to interfere with the board manipulation, we need to place them on the back of the receiver board (on the transmitter board we will be placing it on the middle of the power board).
- **Radio Antennas:** The transmitter antenna does not matter as it is a cable, but for avoiding the receiver antenna interference with the board manipulation is should be placed also on the top layer at the back of the board.
- **ESP-32-WROOM-32D microprocessor:** On the datasheet is indicated how to correctly position this module on a board. We need to pay especial attention to this requirements otherwise it will not work as expected. The main requirement is to place it in such a way that the antenna of this module does not contain any ground plane below it. Beside, there should be some margin space must be left between the antenna and the rest of the modules to avoid interferences (Figure 4.50).

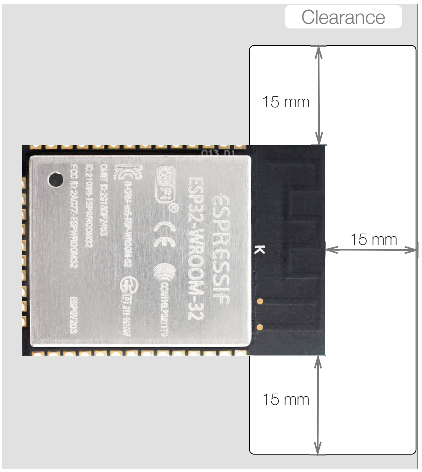


Figure 4.50 – *ESP-32-WROOM-32D recommended antenna clearance [65]*

- **microSD connector:** This connector, analogue to the USB connector must be place somewhere reachable. Therefore, we should e placing it on the board outline.

The final module's placement is exposed in subsubsection 4.1.2.4

4

4.1.2.3 PCB routing

Before finally placing the modules on the board, we need to think of the main routing aspects we need to be aware of for facilitating the routing when placing the modules. Those are the one listed below:

- The track size should be as wide as possible. Therefore we will try to make the digital signal of 20 mils and the power tracks of 30 mils.
- I2C interface modules should be kept closer to the microprocessor I2C pins as we did not include the pull-up resistor thinking of reducing the track length.
- the antennas (radio and GPS) are 50 Ω impedance antennas, therefore the antenna track must have this impedance. For calculating this size we used the **Saturn PCB Design** toolkit as shown in Figure 4.51 and in Figure 4.52.
- The USB tracks are part of a differential pair, therefore they must have the same length

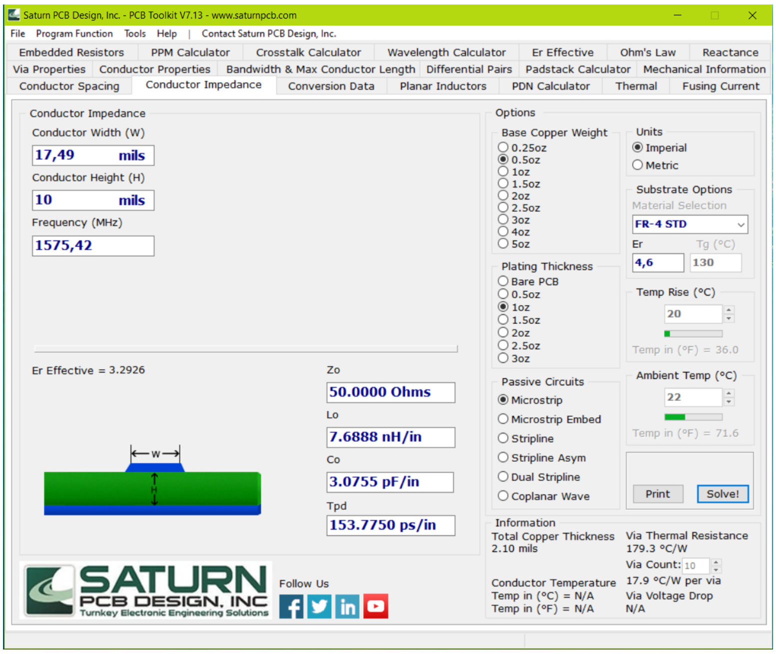


Figure 4.51 – *GPS Antenna track width*

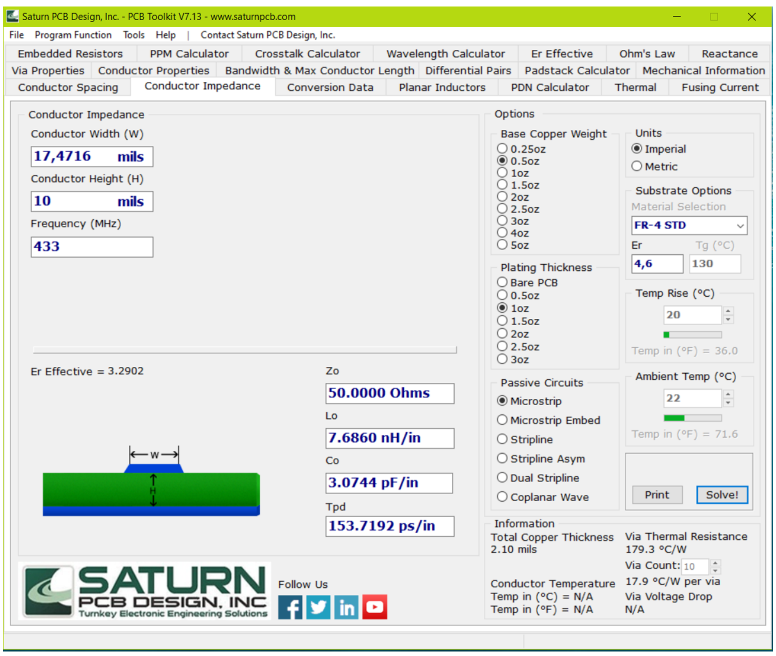


Figure 4.52 – *Radio Module Antenna track width*

Another aspects we must take into account when routing a track is not to track with square tracks, not to track two different tracks in parallel for too long and separate the tracks as much as possible.

4.1.2.4 PCB Final Design

In this section we will be presenting the final design for the four boards we have implemented. For both transmitter and receiver we will be including not only the 3D view of the board from different perspectives but also the 2D PCB prints so that we can have an idea of the tracks we needed to route.

We need to mention that even though we send the PCBs to the *jlcpcb.com* manufacturer, while we waited for the boards to arrive, we decided to manufacture the base board on the laboratory. The reason we only manufactured the base board is just because it was the bigger one which, ideally, will let us solder the modules by hand even though we did not have the solder mask technology. Moreover, we will learn how a PCB machine work and the information we have to proportionate in order to manufacture a PCB. After seeing the PCB manufactures without the solder mask we realized that it was almost impossible to solder any module without short-circuiting any track. The PCB manufactured in the laboratory and a video are included in Subsection 5.1.1.

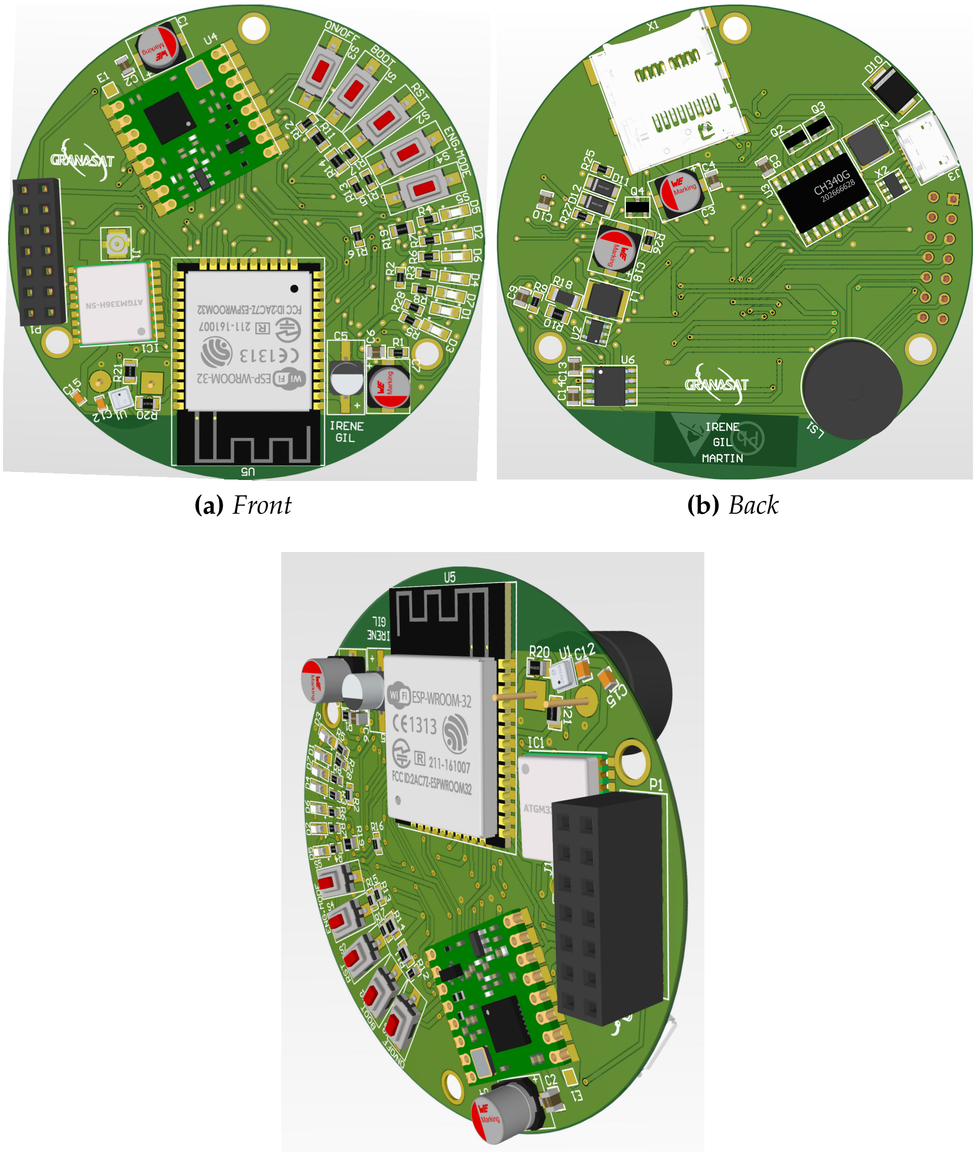
4.1.2.4.1 Transmitter Final Design

In Figure 4.53, Figure 4.54 and in Figure 4.55 we can observe a 3D render of the **Main Board**, **Power Board** and **Uni Board** final design respectively. In those figures are included not only a view from the top and bottom layers but also from a perspective angle to have a visual orientation on how tall the devices are. Whereas in Figure 4.56,Figure 4.57 and Figure 4.58 we have presented a 2D print of those same boards to take a look into the tracks width and separation.

It should be remarked that for the PCB modules' placement we have followed the aspects we listed in subsubsection 4.1.2.2. For that reason we have placed the bus connector, USB connector and buttons on the border outline of the board. Moreover, the microSD has been placed also on the border outline from the bottom layer since we run out of space on the top layer.

The last thing we need to remark is that as we already mentioned in previous sections, the GPS antenna is big enough to take almost one board. Since we wanted to fit all the main modules on the Main Board we decided to move it to the Power Board. Although the module is physically in that module, the connection is made in the Main Board. To do so, the GPS antenna has a cable as we can observe in 4.54c which will be introduced through the hole placed between the bus connector and the antenna and connected to the IPX connector showed in 4.53a under the J1 name.

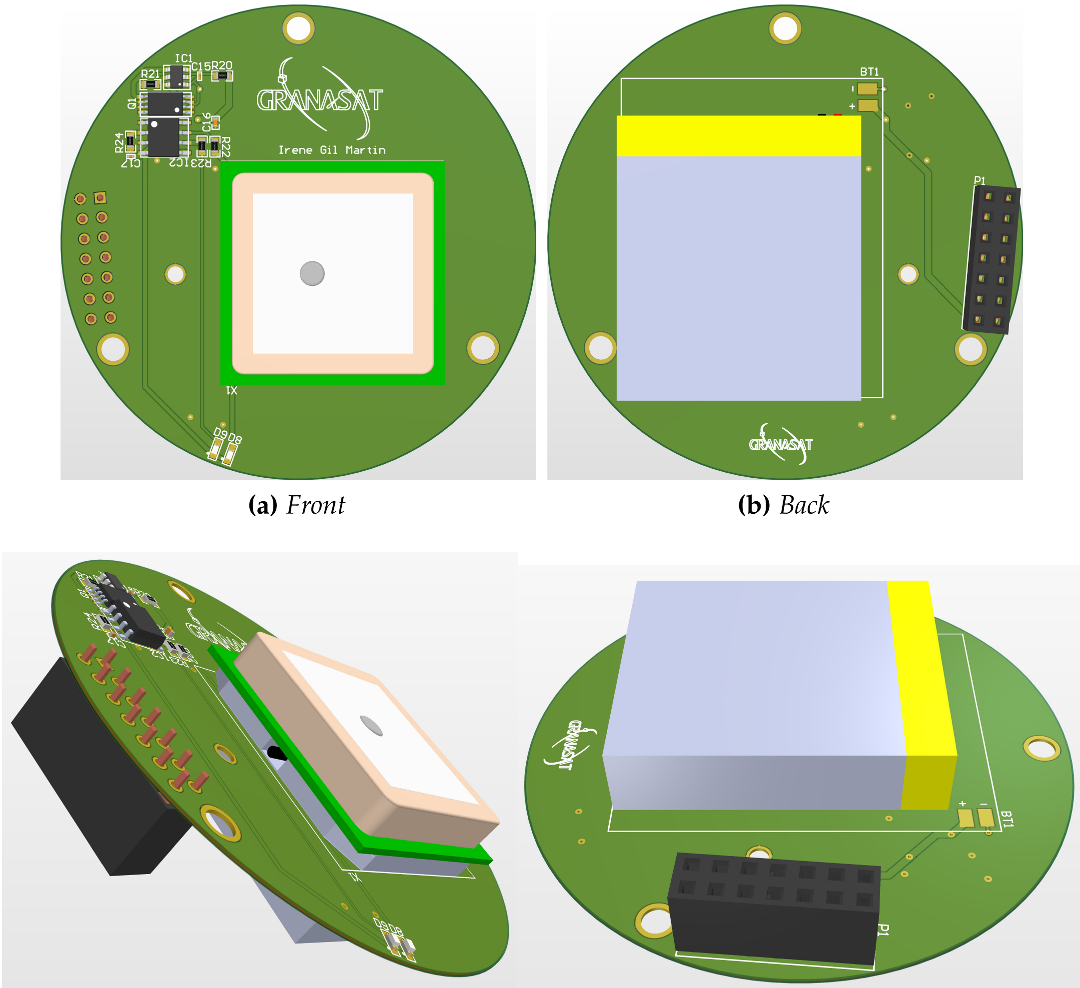
To conclude this section, we need to present the final PCB 3D render that will show how the 3 boards will be connected and placed inside the "can". This final figure can be observed in Figure 4.59.



(a) *Front* **(b)** *Back*

(c) *Side Top Layer*

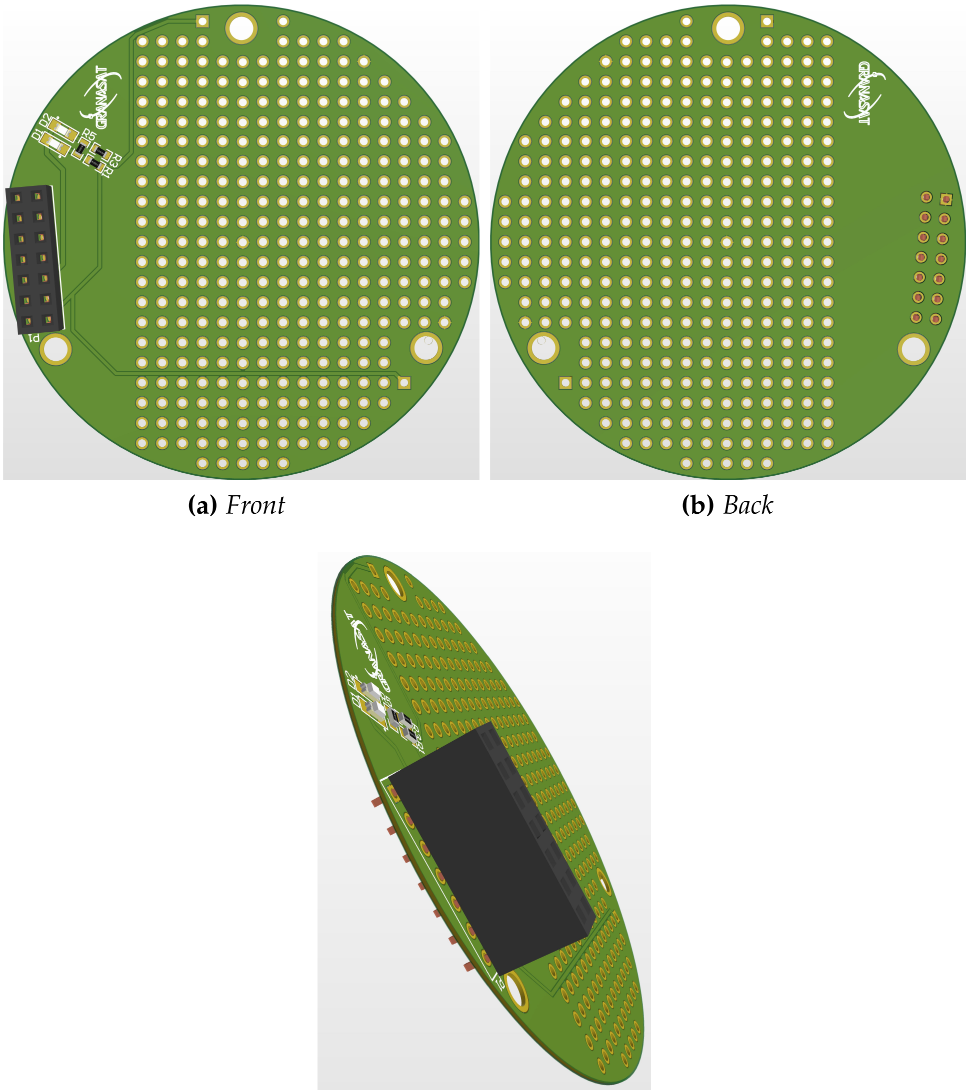
Figure 4.53 – *GranaSAT CanSat Main Board 3D final design*



(a) *Front* **(b)** *Back*

(c) *Side Top Layer* **(d)** *Side Bottom Layer*

Figure 4.54 – *GranaSAT CanSat Power Board 3D final design*



(a) *Front* **(b)** *Back*

(c) *Side Top Layer*

Figure 4.55 – *GranaSAT CanSat Uni Board 3D final design*

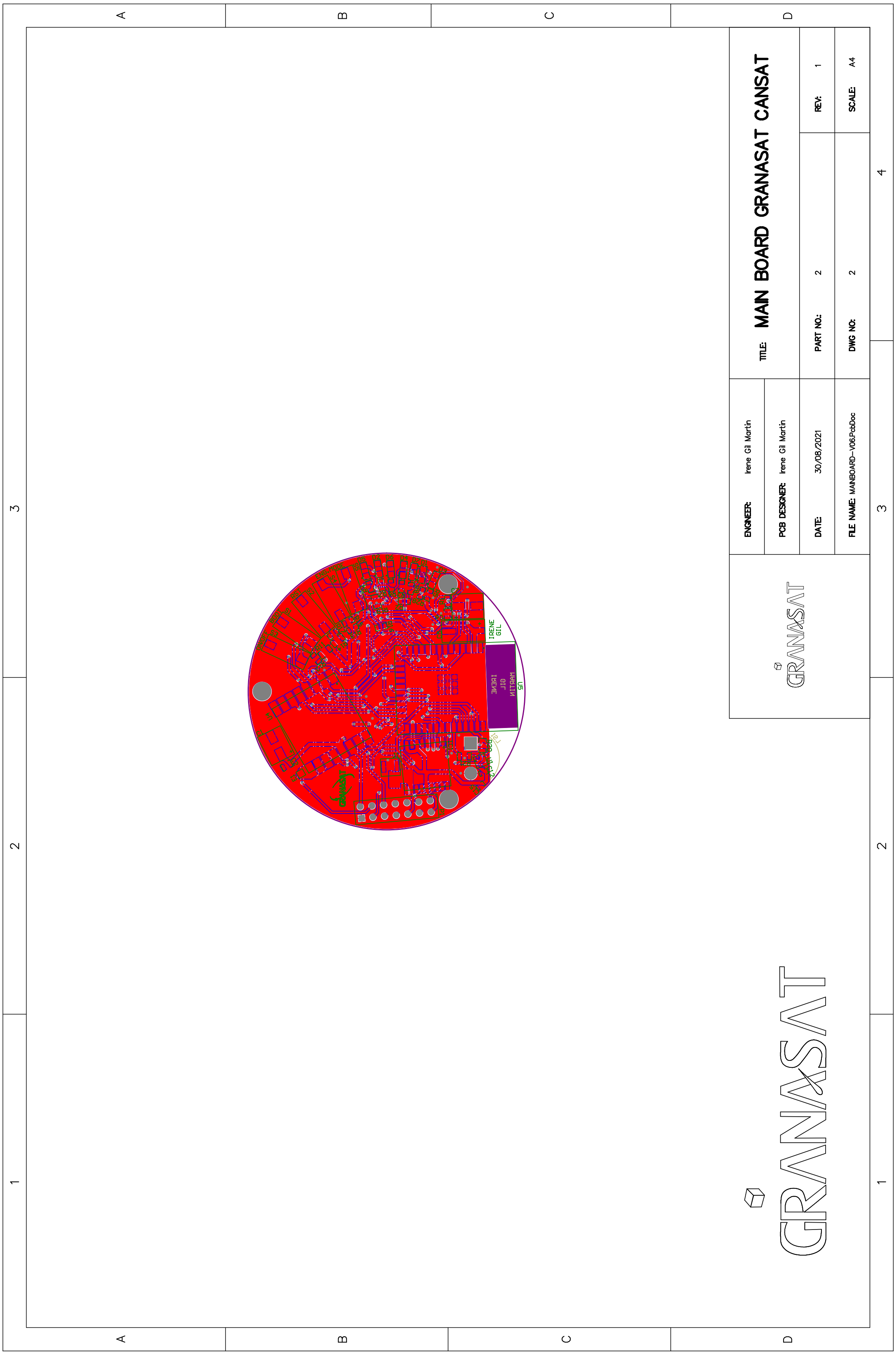


Figure 4.56 – *GranaSAT CanSat Main Board 2D final design*

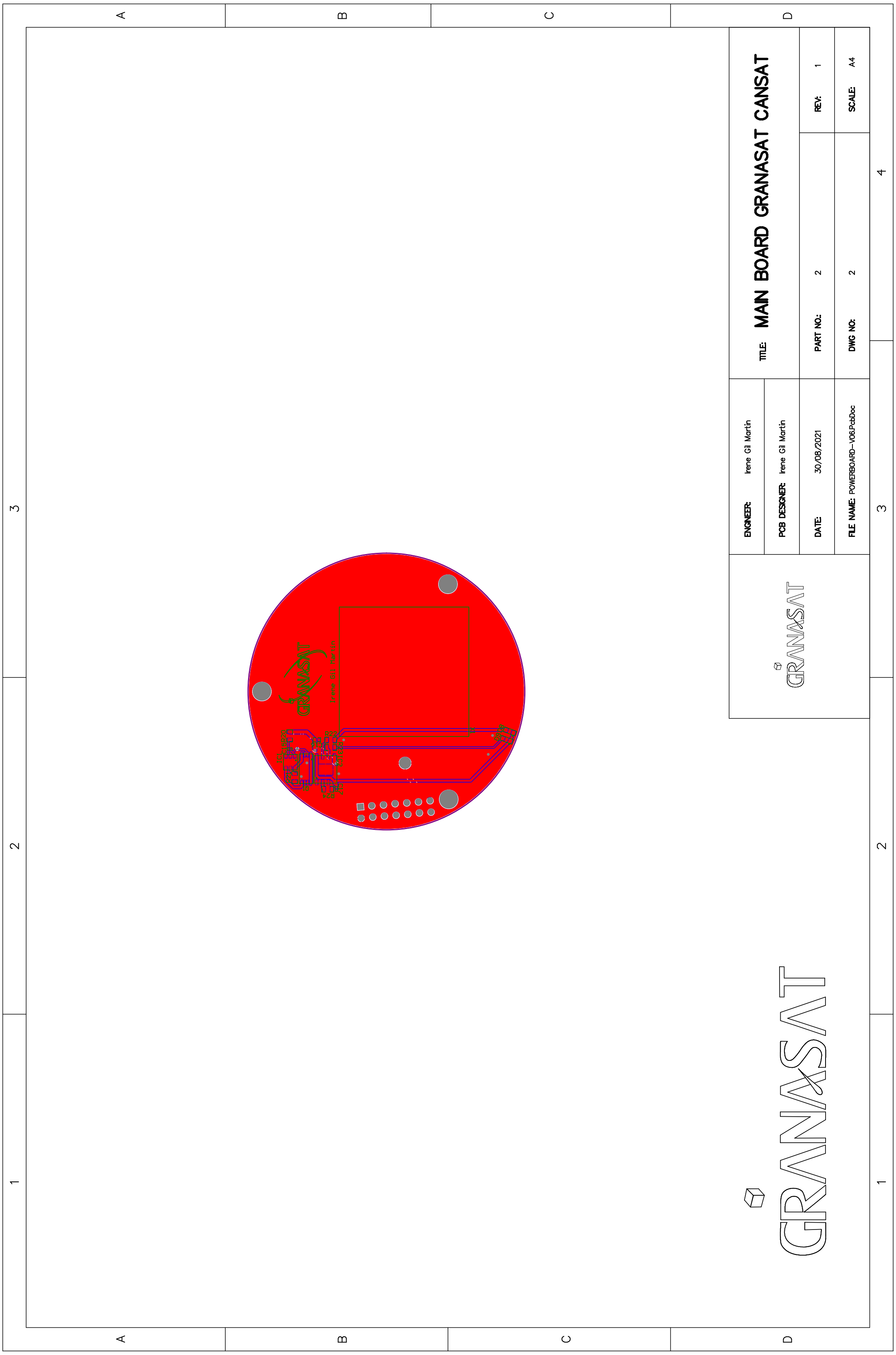


Figure 4.57 – GranaSAT CanSat *Power Board 2D final design*

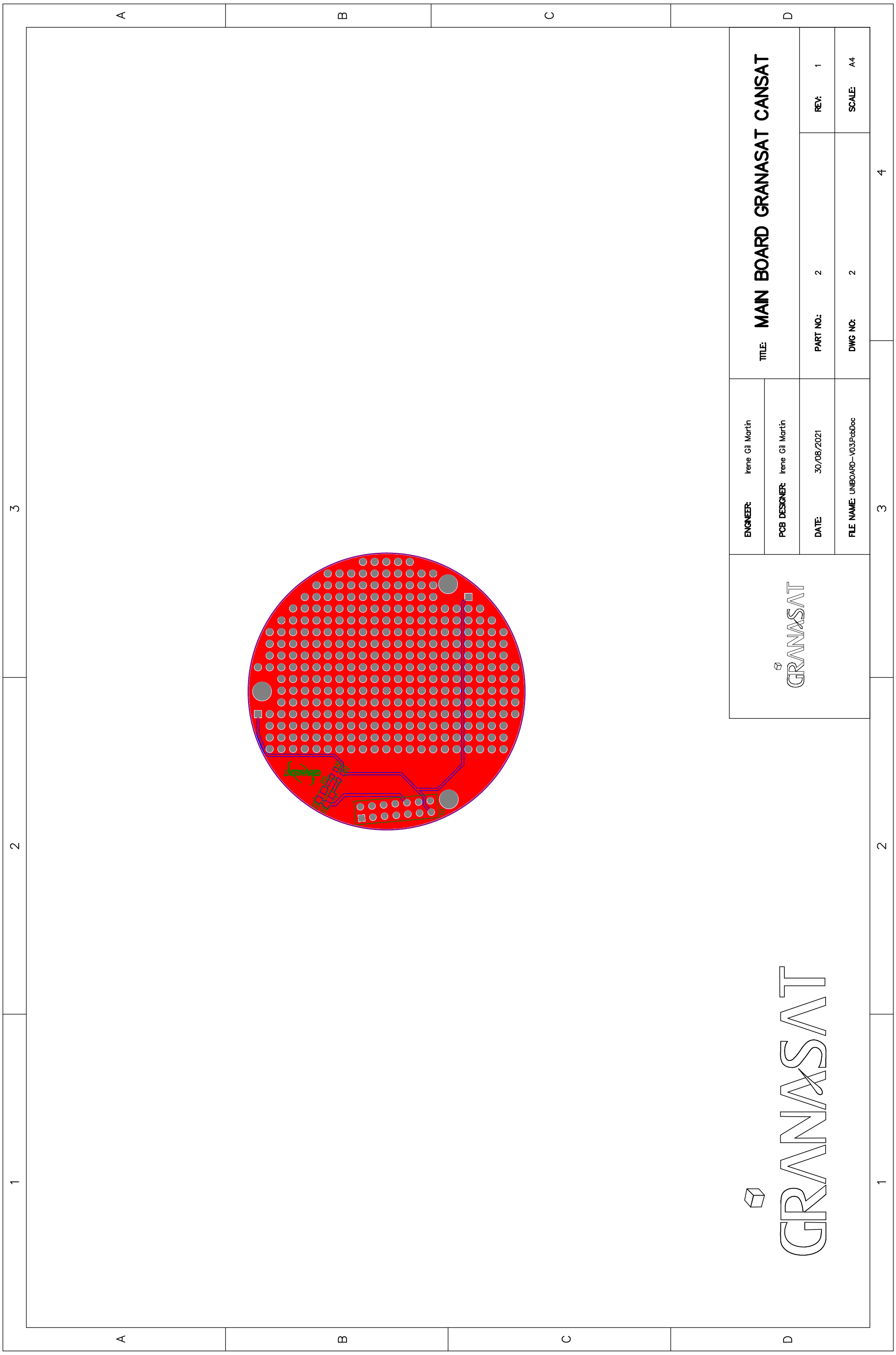
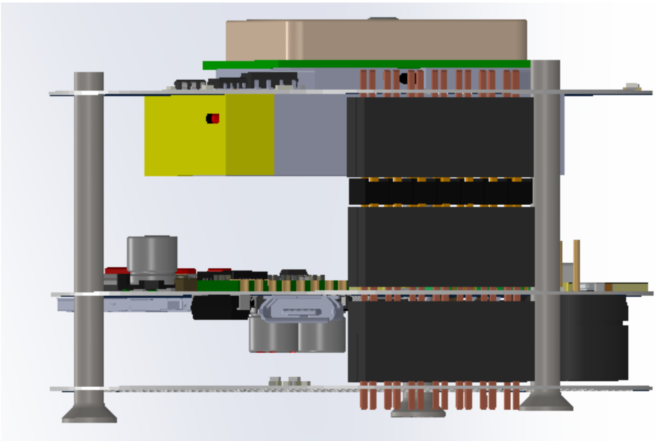


Figure 4.58 – *GranaSAT CanSat Uni Board 2D final design*

4


(a) *Perspective* 1

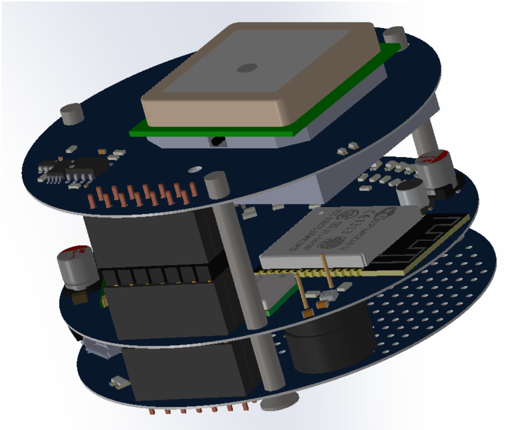

(b) *Perspective* 2

Figure 4.59 – *GranaSAT CanSat transmitter 3D final design*

4

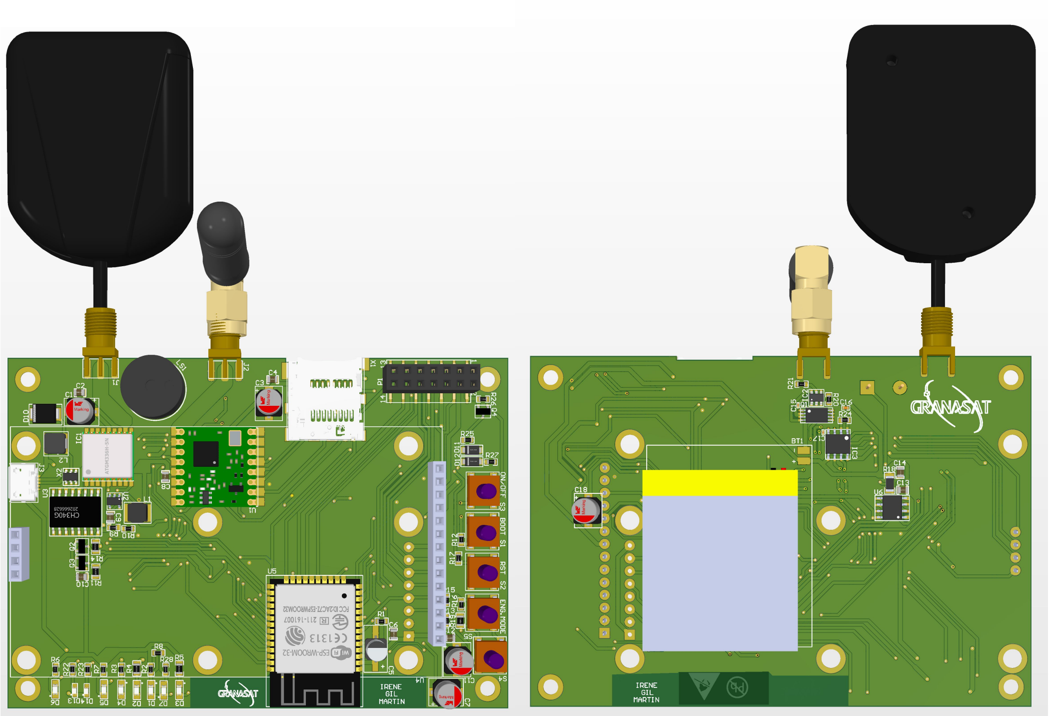
4.1.2.4.2 Receiver Final Design

Analogue to paragraph 4.1.2.4.1, in this section we will be presenting the 3D renders of the **Base Board**. This 3D renders are the one presented in Figure 4.60. Note that in 4.60d we can see the final view with the LCD included. On the other figures, we have changed the LCD transparency so that the rest of the modules that are placed under it can be seen easily.

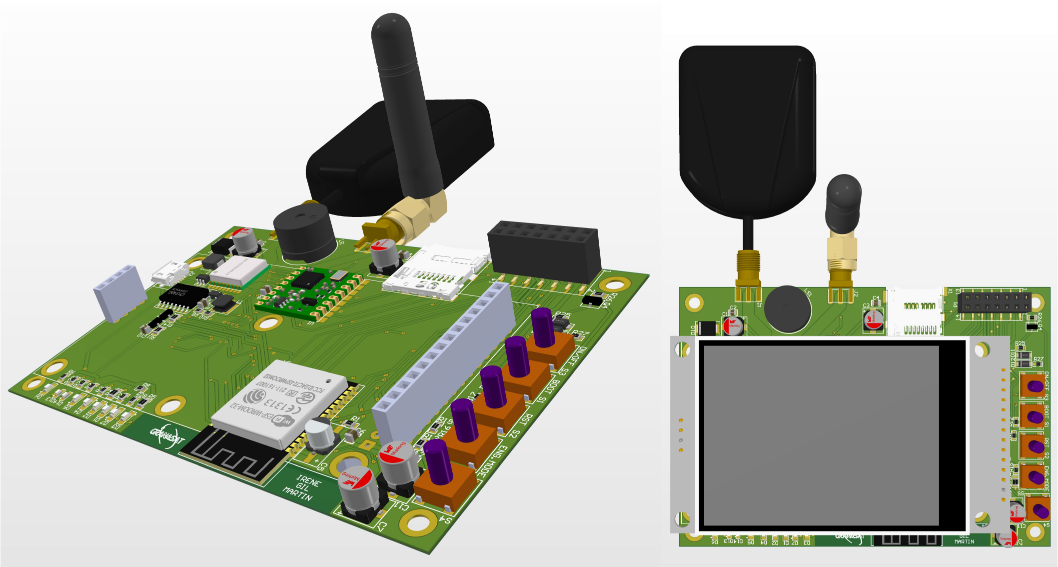
The most important thing we need to highlight is that in this case, for the LCD and LEDs to be visible, the antennas, USB connector, microSD connector and bus header has been sent to the back of the board.

Another thing we need to remark is that for microSD to be accessible we have made a little incursion on the board so that the card is inserted without hitting the board with the fingers. This cut into the board can be perfectly seen in 4.60b.

The last thing we need to comment is that, as we have mentioned in subsubsection 4.1.2.2, the ESP-32-WROOM-32D microprocessor has some especial requirements when placing it on the PCB. This requirements imply that there should be no mass plane under his antenna. Therefore, on both Figure 4.53 and Figure 4.60 we can observe a transparency on that part of the module.



(a) *Front* **(b)** *Back*



(c) *Side Top Layer* **(d)** *Top view with LCD*

Figure 4.60 – *GranaSAT CanSat Base Board 3D final design*

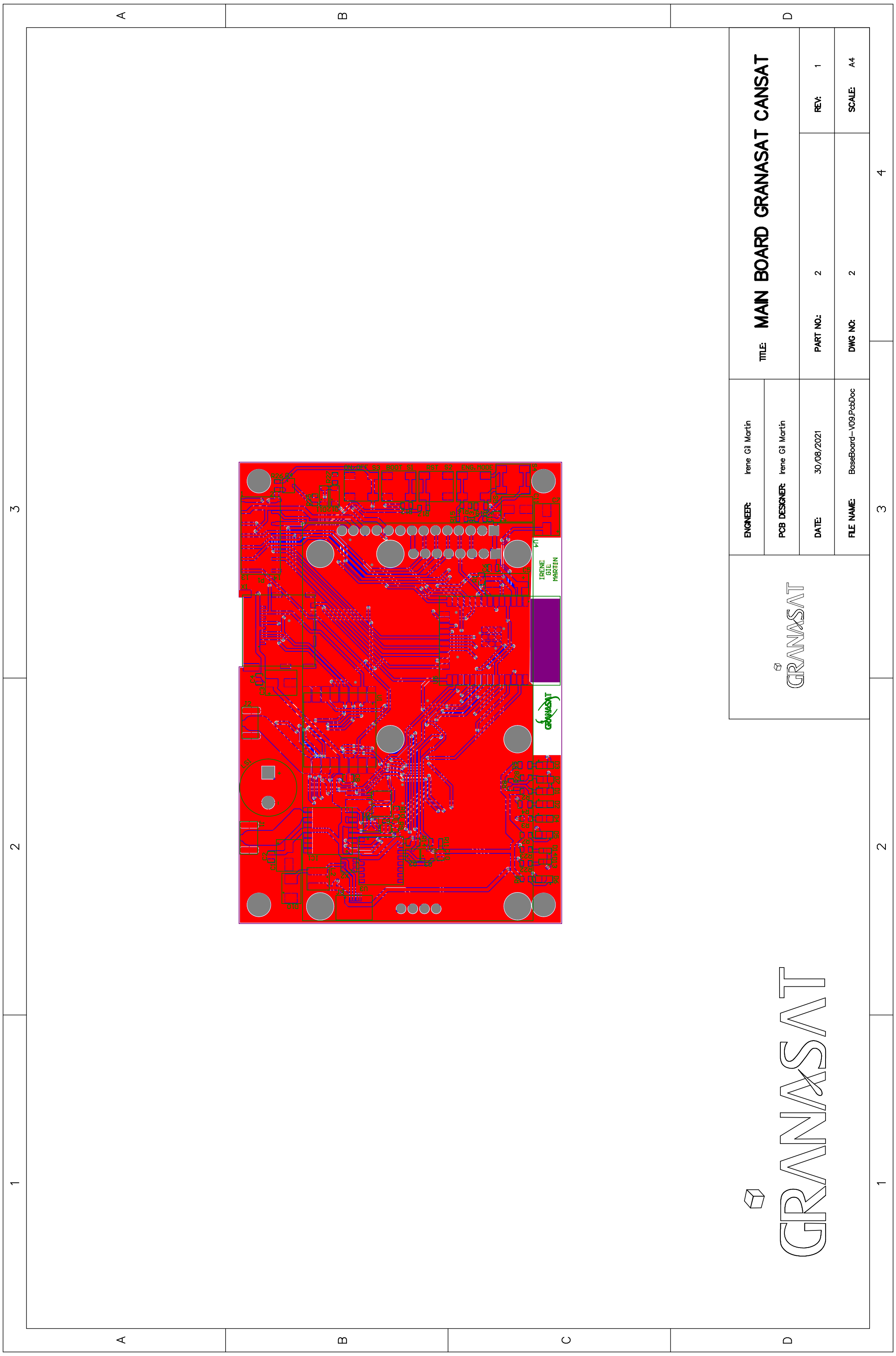


Figure 4.61 – *GranaSAT CanSat Base Board 2D final design*

4.1.3 Power Budget

One of the most important steps needed for a good electronic design is the power budget. Executing a good power budget will give us the best tool for choosing the most suitable battery to supply our device the exact needed time.

Performing a power budget consists of keeping track of all the modules consumption for estimating the device total consumption. To do such thing, we will be using a spreadsheet just like we did in Section 2.2 for the Open CanSat. As we have two different devices, the transmitter and the receiver, we will be performing a power budget for each device.

From the datasheet we will be extracting the typical and maximum current consumption for each module. With that data and the supplying voltage, we will be able to estimate the power consumption (in W) of each module by simply applying Equation 4.1.6.

$$P = \mathrm{V} \cdot \mathrm{I} \tag{4.1.6}$$

The spreadsheet where we calculated each module consumption is the one presented in Figure 4.62.

Component	Module	Number/PCB	V_{IN} (V)	I_{IN_TYP} (mA)	I_{IN_MAX} (mA)	P_{IN_TYP} (W)	P_{IN_MAX} (W)
uProcessor	ESP-32-WROOM32E	1	3,3	500	1000	1,65	3,30
microSD	SanDisk standard uSD	1	3,3	30	100	0,10	0,33
	ATGM-336H	1	3,3	100	100	0,33	0,33
GPS	Ceramic Antenna	1	3,3	10	10	0,03	0,03
	Exterior Antenna	1	3,3	10	10	0,03	0,03
Radio Transmitter	RFM96W	1	3,3	20	120	0,07	0,40
Radio Receiver	RFM96W	1	3,3	10,8	12,1	0,04	0,04
Pressure				2,80E-03	4,20E-03	9,24E-06	1,39E-05
Humidity Sensor	BME280	1	3,3	1,80E-03	2,80E-03	5,94E-06	9,24E-06
Temperature				1,00E-03	2,00E-03	3,30E-06	3,60E-09
Buzzer	KXG1205	1	3,3	10	45	0,03	0,15
Display	ILI9341	1	3,3	80	100	0,26	0,33
USB2TTL	CH340C	1	3,3	7	20	0,02	0,14
Red LED	0805 SMD	1	3,3	1,56	42	0,01	0,14
Blue LED	0805SMD	3	3,3	5,40	258	0,02	0,85
White LED	0805 SMD	3	3,3	4,08	285	0,01	0,94
GREEN LED	0603 SMD	1	3,3	1,00	1,13	0,00	0,004
Red LED	0603 SMD	1	3,3	0,76	0,83	2,49E-03	2,74E-03
DC Current Measurement	INA219A	1	3,3	10	10	0,033	0,033

Figure 4.62 – *GranaSAT CanSat modules consumption*

Once all modules consumption has been estimated, we need to determine the total power and current consumption for each board. To do so, we only need to add up the consumption of all the modules lodged in each board. Note that as we mentioned, we are estimating not only the consumption when the modules are working with the typical consumption, but also the worst scenario, when they are working at its maximum consumption. The battery tension and capacity as well as the total power and current consumptio of each board are exhibit in Figure 4.63.

	TYP	MAX
V_{SOURCE} (V)	3,7	4,2
CAPACITY (mAh)	800	850
SOURCE SPECS	3.7 V / 3.3 V	3.7 V / 3.3 V
TOTAL POWER MAIN BOARD(W)	2,31	6,65
TOTAL POWER BASE BOARD(W)	2,50	6,58
CURRENT NEEDED MAIN BOARD (A)	0,70	1,98
CURRENT NEEDED BASE BOARD (A)	0,76	1,98

Figure 4.63 – *GranaSAT CanSat device's total consumption*

Another important feature to be taken into account when carrying out a power budget is the buck or boost converter efficiency. Since we decided to power up all the modules with 3.3 V, we will only need one buck converter, the one chosen in Subsection 3.2.8. In that section, we compare the LM3671 buck converter with the one we will be using, the AP3429/A. We did not only made the table trade-off from their datasheets but also know the total power this modules can dissipate. As the output current given by the LM3671 buck converter is not enough for our devices, we explore the possibility of including two of them instead of one.

The typical and maximum power and current this modules are able to manage are listed in Figure 4.64.

TYPICAL CONSUMPTION BUDGET					
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT} (A)	P_{IN} (W)
AP3429/A (3.7 V -> 3.3 V)	1	3,7	4,95	1,5	2,96
LM3671 (3.7 V -> 3.3 V)	2	3,7	1,98	0,6	2,96
MAXIMUM CONSUMPTION BUDGET					
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT_MAX} (A)	I_{OUT} (A)
AP3428/A (3.7 V -> 3.3 V)	1	4,2	6,6	2	2
LM3671 (3.7 V -> 3.3 V)	2	4,2	3,795	1,15	1,02

Figure 4.64 – *GranaSAT CanSat buck converter power and current management*

For keeping track of the buck converter power dissipation as well as for the current consumption margin, we will be subtracting the consumption to the values the buck converter is able to supply. The power dissipation is presented in Figure 4.65 whereas the current consumption margin is exhibit in Figure 4.65.

POWER CONSUMPTION	TYP (W)	MAX (W)
AP3429/A_POWER-CONSUMPTION_MB	2,64	-0,05
LM3671_POWER-CONSUMPTION_MB	1,65	0,94
AP3429/A_POWER-CONSUMPTION_BB	2,45	0,02
LM3671_POWER-CONSUMPTION_BB	1,01	1,01

Figure 4.65 – *GranaSAT CanSat buck converter power dissipation*

CURRENT CONSUMPTION	TYP (A)	MAX (A)
AP3429/A_CURRENT-CONSUMPTION_MB	0,80	0,02
LM3671_CURRENT-CONSUMPTION_MB	0,50	0,06
AP3429/A_CURRENTCONSUMPTION_BB	0,74	0,02
LM3671_CURRENT-CONSUMPTION_BB	0,44	0,06

Figure 4.66 – *GranaSAT CanSat buck converter current consumption margin*

We can observe from the Figure 4.65 that in case all modules work wih their maximum consumption, the power dissipation will not be enough with the buck converter we have chosen. But that would be a unlikely scenario that we will not worry about.

From Figure 4.66 we can observe that the current consumption margin we will have for introducing new modules is big enough to not have to worry about this fact.

The last thing we must be aware of is our device autonomy. This is the amount of time our device could be ON without running low on battery. This time is estimated with Equation 2.2.1. Therefore, with our boards consumption and the battery capacity, the autonomy we will have is as presented in Figure 4.67.

	ON TIME (h:min:s)	
MAIN BOARD	1:08:35	0:25:43
POWER BOARD	1:32:29	0:25:43

Figure 4.67 – *GranaSAT CanSat autonomy time estimation*

As we can observer from this image, the device autonomy is more than enough for our application. Within an hour we can synchronize the GPS, place the transmitter into the rocket and perform the launching.

4.1.4 Mass Budget

Going back to Subsection 1.2.1 we will remember the main purpose of our device. This device is intended to be used in the ESA CanSat competition. Consequently, there are a few requirements that our device must met. This requirements were listed in Section 3.1.

In this section we will focus on meeting the mass requirement. This condition gives us a limit of the maximum weight our CanSat transmitter may have. This parameter is set to a maximum of 280 g. Even though this is the maximum value our transmitter can have, we should be approaching this weight since the rocket trajectory will depend on the CanSats weight.

For carrying out the mass budget, we have also used a spreadsheet since is the most comfortable way of doing arithmetical calculations like the ones we are going to be doing. Note that even though we only have a weight restriction in the transmitter, we will also be doing a mass budget for the receiver just for comparison purposes.

In Figure 4.68 we can look up each module weight.

Component	Module	Number/PCB	Mass (mg)
uProcessor	ESP-32-WROOM32E	1	8000,00
microSD	SanDisk standard uSD	1	672,59
GPS	ATGM-336H	1	612,00
	Ceramic Antenna	1	11200,00
	Exterior Antenna	1	79095,00
Radio Transmitter	RFM96W	1	949,00
Radio Receiver	RFM96W	1	949,00
Pressure Humidity Sensor Temperature	BME280	1	65,00
Buzzer	KXG1205	1	1,60
Display	ILI9341	1	36000,00
USB2TTL	CH340C	1	16,70
Red LED	0805 SMD	1	23,00
Blue LED	0805SMD	3	23,00
White LED	0805 SMD	3	23,00
GREEN LED	0603 SMD	1	23,00
Red LED	0603 SMD	1	23,00
DC Current Measurement	INA219A	1	76,00
Buck Converter	AP3429 \A	1	16,70
Battery	Li-Po 550 mAh	1	19950,00
Battery Charger	TP4056	1	77,40
	DW01A	1	33,40
	FS	1	8,00
Pin Header	2x7 pin connector	1	1274
Antenna PCB connector	2 sma right angle connector	2	11334
SMA PCB connector	Antenna SMA connector	1	150,00

Figure 4.68 – *GranaSAT CanSat each module weight*

Therefore, is we add up all the modules we will be lodging on the transmitter and on the receiver,the total weight will be having is the one presented in Figure 4.69. Note that this weight does not include the ”can” extra weight or even the PCB weight. This value only corresponds to the modules total weight.

	TOTAL (mg)	TOTAL (g)
CanSat Transmitter	45857,39	45,86
CanSat Receiver	158231,39	158,23

Figure 4.69 – *GranaSAT CanSat each module weight*

If we want to know the complete transmitter weight counting the PCB and the "can" bulk, we will be referring to Figure 4.70.

	TOTAL (mg)	TOTAL (g)
CanSat Transmitter	45985,39	45,99
CanSat Receiver	158249,39	158,25

Figure 4.70 – *GranaSAT CanSat total weight*

The weight of the PCBs without any modules and the "can" are presented in Figure 4.71.

	Number	Mass (g)
CASE (PLA)	1	56
ROUNDED PCB	3	24
RECTANGLE PCB	1	18

Figure 4.71 – *GranaSAT CanSat PCB and "can" weight*

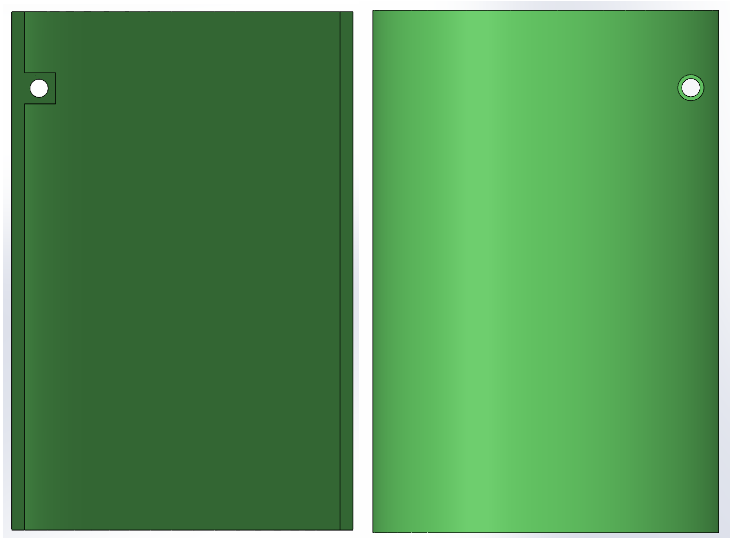
4

4.1.5 CanSat assembly design

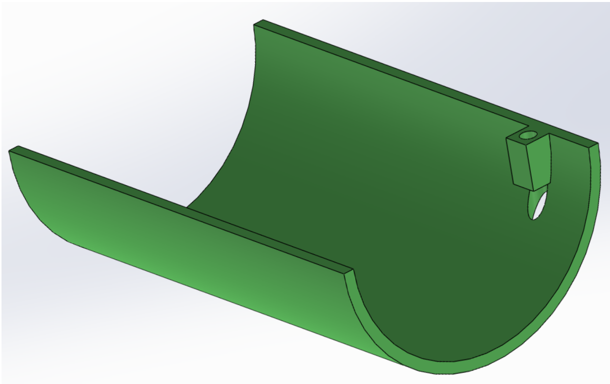
When studying the Czech engineers CanSat design, we studied their "can" model proposal. If we recall from the Section 2.4, this model was not the most optimal since they did not think about opening an space for the GPS antenna or even for the LEDs to be seen. Since the main purpose of this Bachelor Thesis is to improve their design, we have thought of a new model that does implement these aspects.

In Figure 4.74 we can observe two windows we have created for the LEDs to be visible as well as for the buttons to be accessible. As they are placed ones next to the others, this windows should fit perfect for both purposes. Moreover, in those images and in Figure 4.72 we have presented a little hole inside a rectangle thought for adding a screw and a nut to create the perfect closing for the can. With this system, we would not need a rubber for closing the can.

The last thing we need to comment about our design is that in the cover of the can, we have added a hole for the GPS antenna so that the synchronization does not suffer any hindrance. This aspect can be perfectly seen in Figure 4.74. Furthermore, on the base (Figure 4.75), we have added some empty plastic (triangle) to act as a sock-absorbing element for protecting the boards.

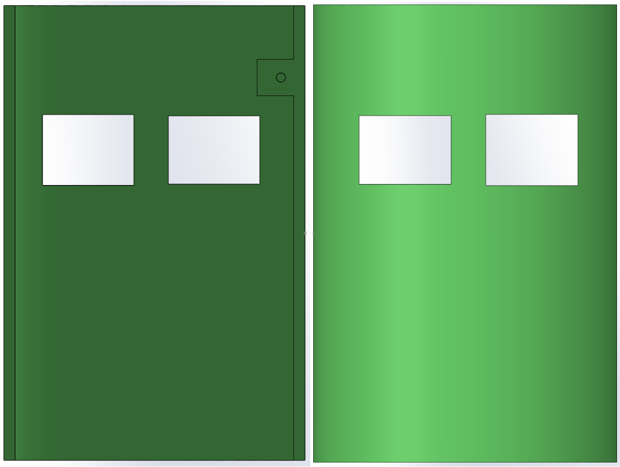


(a) *Front* **(b)** *Back*

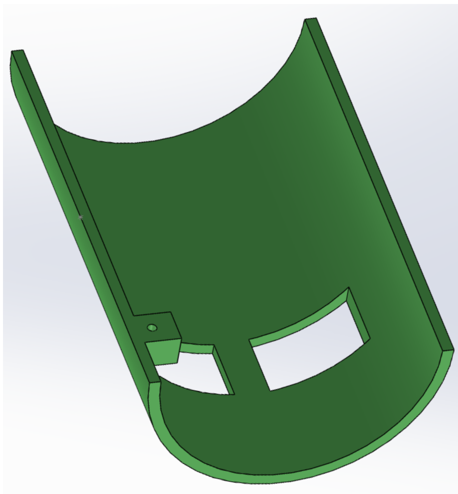


(c) *Side perspective*

Figure 4.72 – *GranaSAT CanSat body "can" design (left)*

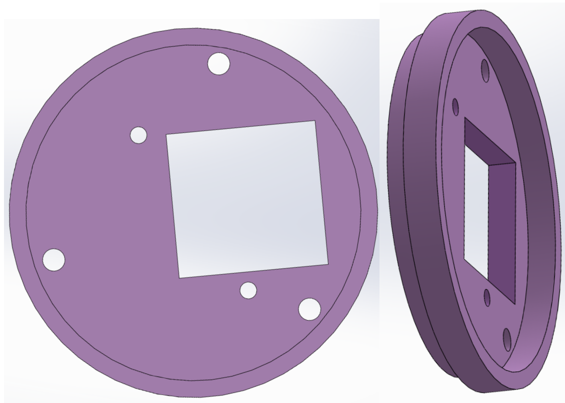


(a) *Front* **(b)** *Back*



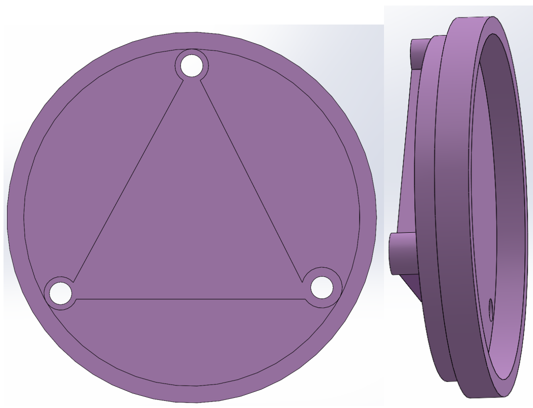
(c) *Side perspective*

Figure 4.73 – *GranaSAT CanSat body "can" design (right)*

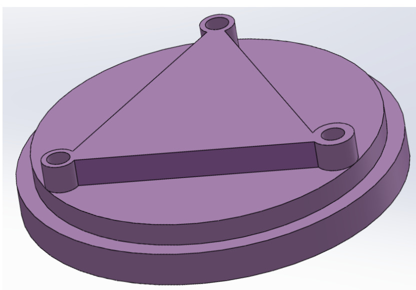


(a) *Front* **(b)** *Side perspective*

Figure 4.74 – GranaSAT CanSat *cover "can" design*



(a) *Front* **(b)** *Side perspective*



(c) *Top perspective*

Figure 4.75 – GranaSAT CanSat *base "can" design*

4.2 Software Design

When designing a new product, first we need to assure that the circuits implemented work individually and then, we can make the device's firmware. Therefore, on this section we will be describing the main commands the modules must receive in order to work. Consequently, we will exposing the examples for each module we have implemented to make sure the design is the correct for the module.

Each module software example has been programmed using Visual Studio Code (VS Code) with the PlatformIO extension as we did on the Czech engineers kit firmware. The reason is the same. Arduino IDE is a much more limiting environment that do not let us organize the project properly as well as have the control over the linking and compiling process. To better understand the project structure of a PlatformIO structure we will refer to Figure 2.34.

Once we have mentioned the platform we will be using, the next thing we should be describing before explaining each module command is the firmware philosophy. Contrary to the **SuperLoop Architecture** we used in the Czech engineers design (Section 2.3), on our design we will be using a RTOS architecture. As this architecture requires more memory than a simple **SuperLoop Architecture**, this is a better architecture to be used in microprocessor such as the ESP32 we will be using. Besides, the two cores of this board will let us divide the task to get a much more compact firmware.

An RTOS is often a lightweight operating system (OS) designed to run on microprocessor. This architecture offers a scheduler to run multiple threads or task,resource management (such as file I/O), and device drivers. This architecture is very useful when running several threads at the same time on a microprocessor and for meeting the time-need since this architecture is able to meet real-time deadlines [53].

To implement this RTOS architecture we will be using FreeRTOS where the main commands we will be using are the ones exhibit in 4.2. All programs developed with FreeRTOS have to include those sentences otherwise they will not work with this architecture.

```
#if CONFIG_FREERTOS_UNICORE
  static const BaseType_t app_cpu = 0; //Setting first core
#else
  static const BaseType_t app_cpu = 1; //Setting second core
#endif

void setup()
{
  xTaskCreatePinnedToCore(Fuction_to_be_called, //Calling the function
                          "TaskName", //Naming the function
                          StackSize, //Saving the Stack Memory that will be used
                          ParamtersToPassToFunction, //Passing parameters needed for the function
                          TaskPriority, // Assigning the priority from 0 to 24 (top)
                          TaskHandle, //Task Handle
                          CoreToUse); //Allocating a Core to run the taks
}

void loop()
{
```

```
    }
```

Code 4.1 – *C++ command used for a basic RTOS program*

It is easy to notice that as we are calling the function on the *xTaskCreatePinnedToCore()* command, this function can be defined on a separate *.cpp* file on the library folder. That way, the main function would be clearer and our project would be more compact. Furthermore, we should notice that the command for running the task has been called on the setup function (which will be executed only once). That is not a problem since in all functions we will be calling a *while (1)* (to run the function forever (*while (TRUE)*)), therefore, the function we will be calling will be running forever.

Once the platform and the philosophy we will be using has been presented, we will be showing the examples we have developed. It should be mentioned that we will be exhibit each module example except of the CanSat complete firmware given that we have only tested the modules on one board (and not all of them has been tested on this board, some of them on the node). Due to this reason and for the lack of time, we will not have the chance of implementing a complete firmware. However, we will be showing a block diagram where we will be presenting what we intended to make as our CanSat firmware.

4.2.1 BME280 Sensor Test Example

Even though we already used this module on the Czech engineers code, we will be changing its code lines. Instead of using global variables for getting their parameters, we will be using pointers in order to reduce execution time as well as memory. Analogue to the code implemented on the Open CanSat kit, the I2C address for this modules is the 0x77 (depending on the hardware connection it could be 0x77 or 0x76).

The code lines we implemented on the function related to this module are exposed in 4.2.1.

```
bme_temp->printSensorDetails(); //Printing the details about the temperature, pressure and humidity accuracy
  //Getting the temperature
bme_pressure->printSensorDetails(); //Getting the Pressure
bme_humidity->printSensorDetails(); //Getting the Humidity
```

Code 4.2 – *C++ code lines for getting the BME280 parameters*

Finally, the code implemented on the *setup function* for calling this function is the one presented in 4.2.1.

```

  xTaskCreatePinnedToCore(BME280_MEASURING, //Calling the function
                          "BME280 MESUARING TASK", //Naming the function
                          1524, //Saving the Stack Memory that will be used
                          NULL, //Passing parameters needed for the function
                          21, // Assigning the priority from 0 to 24 (top)
                          NULL, //Task Handle
                          0); //Allocating a Core to run the task
```

Code 4.3 – *C++ command used for calling the BME280 function in RTOS*

Note that we have assigned this module a 21 priority since this module is needed for achieving the **primary mission**. We have also assigned this task a stack memory of 1524 since we want to print the data on the Serial Monitor and the *Serial.println()* requires some memory. Besides that, the variables declared for saving the data has also a memory requirements.

For closing this subsection we will have to mention that this module has been tested on the node instead of in our design due to the desoldering complications.

4.2.2 Buttons Test Example

To check whether the analogue button circuit designed in paragraph 4.1.1.1.13 worked as expected as well as to check whether the button implemented for powering ON/OFF the device is functional, we decided to run a simple example test. The function created for the analogue buttons is the one presented in 4.2.2. The much more simple function developed for testing the power ON/OFF button is exhibit in 4.2.2.


```

while(1) //Loop for repeating the task forever
 {

  S4_S5_val_get = analogRead(ENG_MODE); //Getting the value of the pin at this moment
    Serial.println("S4 ADC VALUE = " + (String) S4_S5_val_get + " bit"); //Printing the value
  S4_S5_voltage_get = (S4_S5_val_get * 3.3)/ESP32_12_BIT_RESOLUTION; //Getting the bits corresponding to the value of the pin

  if(S4_S5_val_get >= 2200 && S4_S5_val_get <= 2640)    //BUTTON1 pressed

  {
    #ifdef SERIAL_PRINT_DEBUG //Sentence for avoiding wasting time on printing on the Serial Monitor
                              // if the serial communication is not available
        Serial.println("S4 BUTTON IS PRESSED"); //Printing the button has been pressed
        Serial.println("S4 ADC VALUE = " + (String) S4_S5_val_get + " bit"); //Showing the bits
        Serial.println("Voltage on pin when S4 pressed = " + (String) S4_S5_voltage_get + " V"); //printing the voltage
        Serial.println();
    #endif
  }
  else if(S4_S5_val_get >= 1120 && S4_S5_val_get <= 1190) //BUTTON2 pressed
  {
    #ifdef SERIAL_PRINT_DEBUG//Sentence for avoiding wasting time on printing on the Serial Monitor
                              // if the serial communication is not available
        Serial.println("S5 BUTTON IS PRESSED");//Printing the button has been pressed
        Serial.println("S5 ADC VALUE = " + (String) S4_S5_val_get + " bit"); //Showing the bits
        Serial.println("Voltage on pin when S5 pressed = "+ (String) S4_S5_voltage_get + " V");//printing the voltage
        Serial.println();
    #endif
  }
  else if(S4_S5_val_get >= 1790 && S4_S5_val_get <= 1900) //BUTTON1 and BUTTON2 pressed together

  {
    #ifdef SERIAL_PRINT_DEBUG//Sentence for avoiding wasting time on printing on the Serial Monitor
                              // if the serial communication is not available
        Serial.println("S4 AND S5 BUTTONS ARE PRESSED");//Printing the button has been pressed
        Serial.println("S4 and S5 pressed ADC VALUE = "+ (String) S4_S5_val_get + " bit"); //Showing the bits
        Serial.println("Voltage on pin when S4 and S5 pressed = "+ (String) S4_S5_voltage_get + " V");//printing the voltage
        Serial.println();
    #endif
  }
```

Code 4.4 – *C++ command used for the analogue pin Buttons*

```
state = digitalRead (ON_OFF_LECTURE); //Reading the state of the button (HIGH OR LOW)

    if(state!=lastBtnState) //If the value taken is different that the value taken on the last measure
    {
      lastBtnState = state; //Changing the value of the state
```

```
    if (state == LOW) //If the value taken is that the Button is pressed (LOW EFFECTIVELY)
    {
      Serial.println("ON OFF LECTURE BUTTON IS PRESSED!"); //Displaying that the button is pressed
      Serial.println();
    }

  }
```

Code 4.5 – *C++ command used for the digital pin Button*

Finally, the task-calling in the RTOS way for both functions is presented in 4.2.2.

```

  xTaskCreatePinnedToCore(on_off_lecture,   //Calling the task to be executed
                          "on_off_lecture", //Task name
                          3024,             //Stack size
                          NULL,             //Parameter to pass to function
                          3,                //Task Priority (top priority = 24)
                          NULL,             //Task_handle
                          1);               //Core that will execute the task (0 or 1)

  xTaskCreatePinnedToCore(eng_mode,   //Calling the task to be executed
                          "eng_mode buttons lecture", //Task name
                          4024,             //Stack size large as it needs some variables
                          NULL,             //Parameter to pass to function
                          15,                //Task Priority (top priority = 24)
                          NULL,             //Task_handle
                          1);               //Core that will execute the task (0 or 1)
}
```

Code 4.6 – *C++ command used for calling the Buttons functions in RTOS*

In this case, the analogue pin buttons has been assigned with a priority of 15. This buttons will be used as engineering mode setting, therefore they are not as important for the CanSat requirements as the sensor. On the other hand, the power ON/OFF button has been assigned with a priority of 3 since when this buttons will be pressed, we intend to suspend the rest of the task for powering ON or OFF the device.

It should be noted that, for reading the value of the pins, the should be declared as exhibit below:

```

  pinMode(ON_OFF_LECTURE, INPUT); //Setting the ON_OFF_LECTURE PIN AS INPUT
  pinMode(ENG_MODE, INPUT); //Setting the ENG_MODE PIN AS INPUT
```

Code 4.7 – *C++ command used for setting both pins as INPUTS*

4.2.3 GPS Test Example

On the Czech engineers kit, this module was interfaced through I2C interface therefore we only needed to use the *Wire.h* library. On this case, this module is interfaced with a software UART (paragraph 4.1.1.1.5), therefore we will need to include the *SoftwareSerial.h* library. Once this library is included, the code lines we will need to add for receiving the GPS strings are presented in 4.2.3. Note that in order to work properly, first we have to initiate the serial GPS communication as shown in 4.2.3

```
  while(gps.available() > 0) //Checking if there is any byte available on the Serial BUS
    {
    #define GPS_BAUDRATE 9600 //Line included in define.h
    gps.begin(GPS_BAUDRATE);    //Initiating GPS (Software Serial Communication) in the setup function
```

Code 4.8 – *C++ code lines for getting the GPS strings*

```
while(gps.available() > 0) //Checking if there is any byte available on the Serial BUS
    {
      char c = gps.read();
      #ifdef SERIAL_PRINT_DEBUG
        Serial.print(c); //Printing the bytes received if there is a byte available
      #endif
      /*
        dato = gps.read();
        #ifdef SERIAL_PRINT_DEBUG
          Serial.print(dato);
        #endif
      */
    }
      #ifdef SERIAL_PRINT_DEBUG
        Serial.println(); //Printing the bytes received if there is a byte available
      #endif
```

Code 4.9 – *C++ code lines for getting the GPS strings*

Consequently, the RTOS function calling will be included as 4.2.3.

```

xTaskCreatePinnedToCore(gps_lecture,   //Calling the task to be executed
                        "gps lecture", //Task name
                        2824,            //Stack size
                        NULL,            //Parameter to pass to function
                        20,              //Task Priority (top priority = 24)
                        NULL,            //Task_handle
                        1);              //Core that will execute the task (0 or 1)

}
}
```

Code 4.10 – *C++ command used for calling the GPS function in RTOS*

The priority of this task will be less than the BME280 priority but higher than the engineering mode buttons since this module will help us locate the CanSat on the landing.

4.2.4 INA219 Test Example

Since this module is the same we have used in the Czech engineers code, we will be refering to subparagraph 2.3.1.0.2.1 for seeing the main commands. The I2C address is the same as we used then (0x40).

We have not implemented a test program for this module as we make a mistake in this module's footprint, but we intended to assign a 19 priority to this module. Less than the one assigned to the GPS but higher than the one of the engineering mode buttons.

4.2.5 LEDs Test Example

Checking whether the LEDs work as expected or not is a simple program where we set each LED HIGH and LOW for a certain time. The first thing we need to do is to set every LED pin as an output as presented in 4.2.5.

```

pinMode(LED_PIN, OUTPUT); //Setting the ON_OFF_LECTURE PIN AS INPUT
```

Code 4.11 – *C++ command used for setting a LED pin as OUTPUT*

The we will need to create a new function on the library for setting this pin HIGH or LOW for a certain time as exhibit in 4.2.5.

```
while(gps.available() > 0) //Checking if there is any byte available on the Serial BUS
    {
    digitalWrite(LED_PIN, HIGH);
    vTaskDelay(1000/portTICK_PERIOD_MS);  //Delaying 1 second before setting led pin low
    digitalWrite(LED_PIN, LOW);
    vTaskDelay(1000/portTICK_PERIOD_MS);  //Delaying 1 second before setting led pin low
```

Code 4.12 – *C++ code lines for getting blinking the LEDs*

The last thing we need to do is to call the RTOS command line for executing the function. To do such thing we will need to implement the code lines presented in 4.2.5.

```

  xTaskCreatePinnedToCore(leds_test,   //Calling the task to be executed
                          "LEDs TEST", //Task name
                          1524,            //Stack size
                          NULL,            //Parameter to pass to function
                          13,              //Task Priority (top priority = 24)
                          NULL,            //Task_handle
                          1);              //Core that will execute the task (0 or 1)

}
```

Code 4.13 – *C++ command used for calling the GPS function in RTOS*

In this case, as the visual indications are a secondary thing, we have set the task priority as a 13 since we believe is more important to configure the engineering mode than to see the LEDs blinking.

4.2.6 Buzzer Test Example

Analogue to the LEDs function, this auditive indicator is a secondary thing although more important than the visual indications. Therefore this taks will be assigned a priority of 14. Consequently, the RTOS routine we will need to implement is the one presented in 4.2.6.

```

  xTaskCreatePinnedToCore(buzzer,              //Calling the task to be executed
                          "buzzer",          //Task name
                          1024,              //Stack size
                          NULL,              //Parameter to pass to function
                          14,                 //Task Priority (top priority = 24)
                          NULL,              //Task_handle
                          1);                //Core that will exute the task (0 or 1)
```

Code 4.14 – *C++ command used for calling the Buzzer function in RTOS*

Before initiating the commands for the buzzer function, we will need to initate this element by setting the frequency, the channel, resolution and pin. Therefore we will need to add before the 4.2.6 the following lines:

```

ledcSetup(channel_buzzer, freq_buzzer, resolution_buzzer); //channel = 0 (via PWM), freq=2400,resolution=8
ledcAttachPin(buzzer_pin, channel_buzzer); // pin=4,duty cycle=125 (50%)
```

Code 4.15 – *C++ command used for setting up the Buzzer*

Once the RTOS function calling has been defined as well as the initial configuration of the element, we will need to present the main code lines for making this module work. This code lines are the one exhibit in 4.2.6.

```

    ledcWriteTone(channel_buzzer, freq_buzzer);    //Setting the duty cycle for the tone
    ledcWrite(channel_buzzer, duty_c_buzzer_on);   //Emitting the sound
    vTaskDelay(300/portTICK_PERIOD_MS);            //Delaying the sound 300 ms
    ledcWrite(channel_buzzer, duty_c_buzzer_off); //Shutting off the buzzer sound
    vTaskDelay(1000/portTICK_PERIOD_MS);           //Waiting 1 second before starting again
```

Code 4.16 – C++ *command used for making the Buzzer sound*

4.2.7 Radio RF96 Test Example

Since we have only implemented a test example code, we will not be sending any variables. In this case we will be sending a string with a ”Hello Word” sentence. They way we intend to send the variables is the same as the one we used in subparagraph 2.3.1.0.2.4 when sending the variables in the Czech engineers code.

The first thing we need to do for this module before sending or receiving any string is to set the basic parameters. This parameters are set in the *setup function()* with the code lines exhibit in 4.2.7.

```

    !rf96.setFrequency(RF96W_FREQUENCY) //RF96W_FREQUENCY = 433 MHz
    rf96.setModeTx(); //Setting the register where the transmitter will be writing
                      //If we were programming the receiver we will need to substitute
                      //this line with setModeRx()
    rf96.setTxPower(23, false); //Setting the transmitting power in dB. The second parameter correspond
                                //to the PA_BOOST or RFO module pin we are using

}
```

Code 4.17 – C++ *command used for setting the radio parameters*

Note that this parameters are explicitly changed for the ones by default set when the radio module is initiated. Also, when initiating the module we have set some configurations by default such as the FIFO settings, the preamble length (8 bytes) and the GFSK modulation (Rb = 250 kbs and Freq = 250 kHz). This parameters could also be changed but, since this is only an example test, we will be leaving them as the parameters by default.

The code lines for sending an string and receiving the corresponding receiving answer are exhibit in 4.2.7.

```

    char radiopacket[20] = "Hello World #      ";
    itoa(packetnum++, radiopacket + 13, 10); //CONVERTING AN INTEGER TO A STRING: => VALUE   TO
                                               CONVERT:PACKET_NUMBER INTO A STRING,
                                             //RADIOPACKET ARRAY IN MEMORY WHERE TO STORE THE RESULTING NULL-
                                               TERMINATED STRING
                                             //10 BASE- NUMERICAL BASE USED TO REPRESENT THE VALUE AS A STRING--
                                               DECIMALBASE = '10'
    rf96.send((uint8_t *)radiopacket, 20); //Sending the string indicating the length of the message (20)
    rf96.waitPacketSent(); //Wait for any previous transmit to finish


    rf96.recv(buf, &len);
```

Code 4.18 – C++ *command used for sending a string with the RF96 module*

At this point, the last thing we need to do is to call the RTOS commands for executing this module function. Consequently, the code lines needed are the one presented in 4.2.7.

```

    xTaskCreatePinnedToCore(rf96_transmitter,   //Calling the task to be executed
                      "rf96_transmitter", //Task name
                      1924,            //Stack size
                      NULL,            //Parameter to pass to function
                      20,               //Task Priority (top priority = 24)
                      NULL,            //Task_handle
                      0);              //Core that will execute the task (0 or 1)
}
```

Code 4.19 – *C++ command used for executing the RF96 module in RTOS*

Note that even though we have assigned the same priority for this module as we assigned the GPS, in this case, we have assigned the first core to execute this function. We have done such a thing because this is the most critical function our device must met, therefore is better if it is assigned to a free core for not interfering with the rest of the functions.

It also should be noted that this module has been tested on the node since we experimented some problems with the HSPI interface on the microprocessor. Therefore we could only test the transmitter instead of testing the receiver as well.

4.2.8 SD Test Example

This module has also be tested in the node since we experimented some problems not only with the HSPI interface on this microprocessor but also with the soldering part (difficult to solder by hand due to its internal pads). Even though we have tested it on the node, we also experimented some troubles with the breakout board we used. The code lines we implemented for this module are the one presented in 4.2.8.

,

```

    File file = SD.open("/BABA.txt"); //Opening/Creating the file
    writeFile(SD, "/helicoptero.txt", "Reading ID, Date, Hour, Temperature \r\n");   //Wrinting in Helicopter.txt
     a message
    file.close(); //closing the files
```

Code 4.20 – *C++ command used for creating a file*

The lines for calling the RTOS executing function routine are presented in 4.2.8.

```

    xTaskCreatePinnedToCore(uSD_firmware,   //Calling the task to be executed
                      "uSD_firmware", //Task name
                      4824,            //Stack size
                      NULL,            //Parameter to pass to function
                      22,               //Task Priority (top priority = 24)
                      NULL,            //Task_handle
                      1);              //Core that will exute the task (0 or 1)
}
```

Code 4.21 – *C++ command used for writing in the microSD in RTOS*

The priority we have assigned to this module is 22 since we think is very important to store the data measured in case the radio communication does not go as planned. If we lose the data measured, we will not have anything for doing the posterior data presentation.

4.2.9 LCD Test Example

The last module we have tested is the LCD. As we have not solder any module into the Base Board due to the missing mass plane, we tested this module on the node. It should be said that as the rest of the SPI modules, we have experience some trouble with the HSPI interface on this microprocessor.

For testing this module we have interface both the LCD and the touch sensor with the VSPI interface. We have implemented an example where some numbers are printed on the display and with the touch sensor we will be able to calculate some basics operations (such as additions and subtractions). Since this test is a much more longer test and requires more code lines than the rest of the modules, we will be showing the main commands we have used. This code lines are exhibit in 4.2.9.

4

```

  tft.setRotation(0);
  touch_calibrate();
  tft.fillScreen(TFT_BLACK);
  tft.drawRect(DISP_X, DISP_Y, DISP_W, DISP_H, TFT_WHITE);
 drawKeypad();
  tft.setTouch(calData);
```

Code 4.22 – *C++ command used for writing into the LCD*

For more information regarding this module's code we will refer to the project example included on the corresponding folder to this Bachelor Thesis.

4.2.10 GranaSAT Complete Firmware Diagram

The firmware we have thought would be best for implementing all task in the transmitter is the one exhibit in Figure 4.76. This firmware will follow the same routines as we intended in the Open CanSat kit firmware in Figure 2.33.

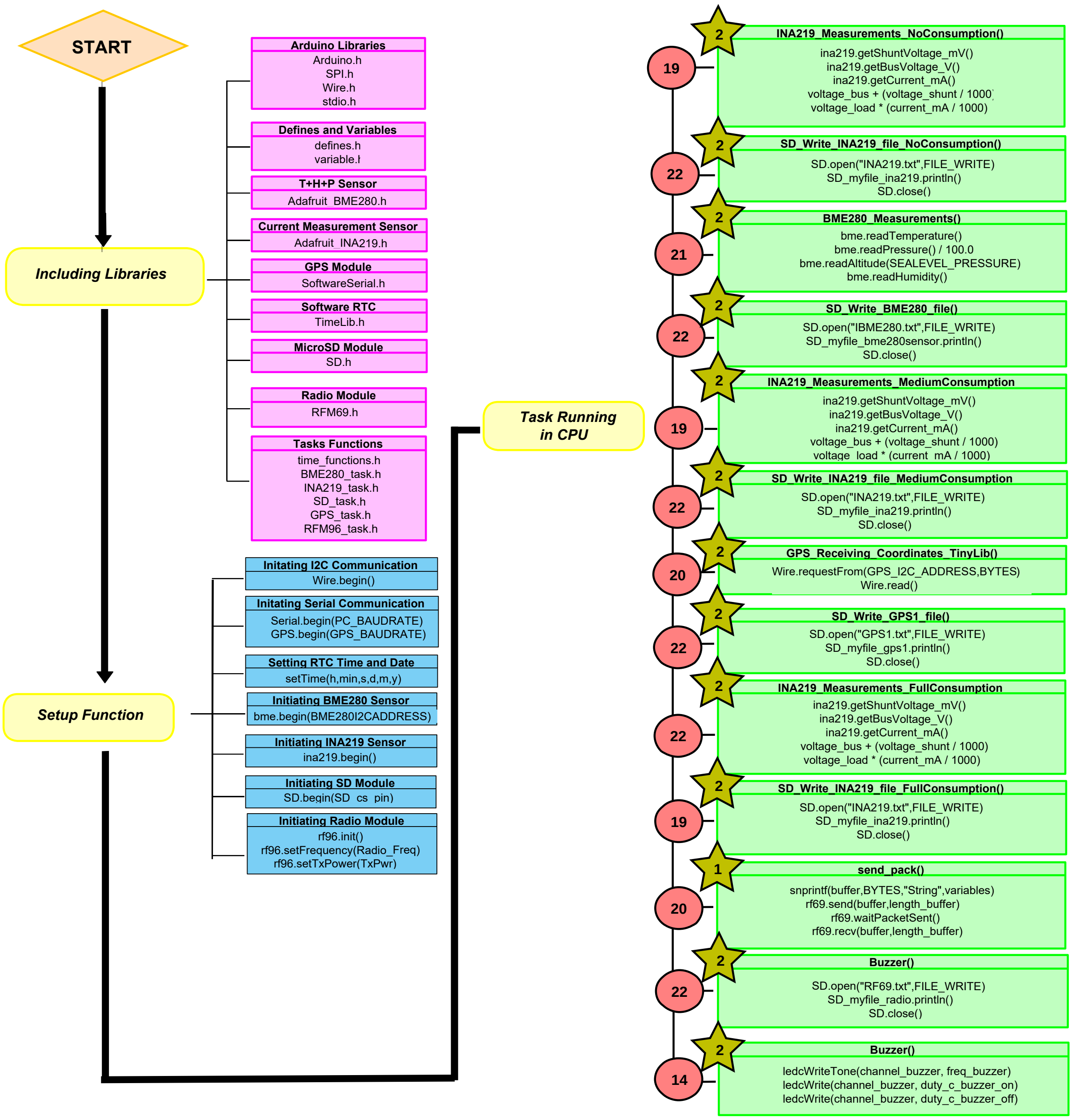


Figure 4.76 – *Block Diagram GranaSAT CanSat Transmitter*

On the image above we could interpret the red circles as the **Task Priority** and the yellow stars as the **Core Assigned**.

However, with this device's transmitter we would like to implement an state machine where the normal program execution would be interrupt each time the power ON/OFF button is pressed. This state machine scheme is the one exhibit in Figure 4.77.

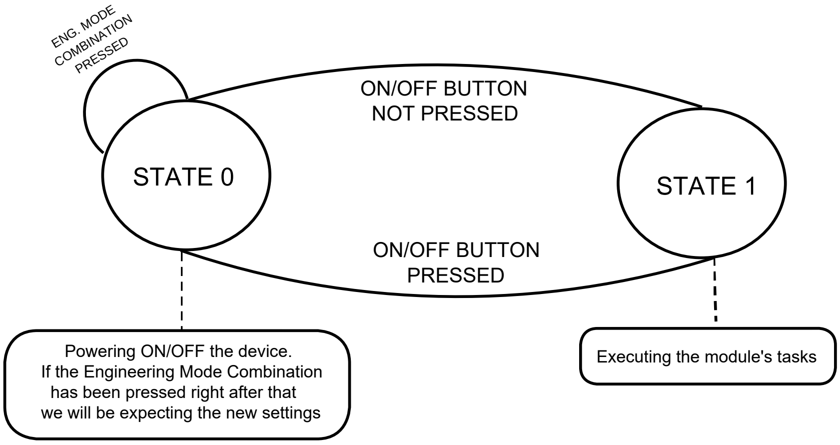


Figure 4.77 – *State Machine GranaSAT CanSat Transmitter*

We have not thought that much about the receiver's complete but we will be implementing most of the tasks of the transmitter, except for the BME280, with the same priorities and core assignment. Besides, on this board firmware we will have to add the LCD. Since we have not managed to make this module work perfectly, we have not thought about the state machine for this firmware. Depending on whether we achieve to make the touch sensor work or not, the state machine will be different. Therefore we will have to wait until we make this module work until deciding the receiver's firmware.

Chapter 5

Integration, Test and Verification

This fifth chapter is used to validate the solutions proposed in this Bachelor's Thesis and check the accomplishment of the goals of the project. This verification is performed using two resources: on the one hand, different photos of the developed products, which complement the ones introduced in each section of the Chapter 4; on the other hand, the firmware (Section 4.2) is used to verify the functioning of the different subsystems, the transmitter and the receiver. This section will also be a basis for the next chapter, Chapter 6, where we will exposed the possible improvements for our design after all tests performed and described in this chapter.

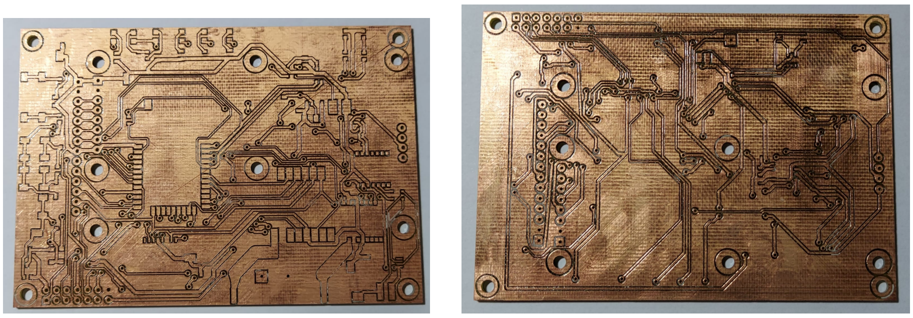
To follow the best structure for that purpose first of all we will be showing the PCBs received from the manufacturer. Then we will show some pictures of the PCBs solder and last of all, we will be showing some pictures and captures from the test performed to check the boards utility. This captures will show not only the captures from the serial monitor, but also the oscilloscope images with the signals.

5.1 Final product

In this section, different photos of the **GranaSAT CanSat** redesign are shown. Also we will briefly comment on the mistakes made.

5.1.1 PCB Manufactured in the GranaSAT Laboratory

As we already mentioned in paragraph 4.1.2.4.1, while we waited for receiving the PCBs manufactured in *jlcpcb.com*, we decided to learn the process of fabricating a PCB by manufacturing the Base Board (bigger one) in the laboratory. The result of this fabrication is presented in Figure 5.1. Once the PCB was fabricated, we realized how hard would be to solder the elements by hand without having a solder mask. The risk of making a short-circuit were too high. Therefore, we decided to wait until the PCBs order was received.



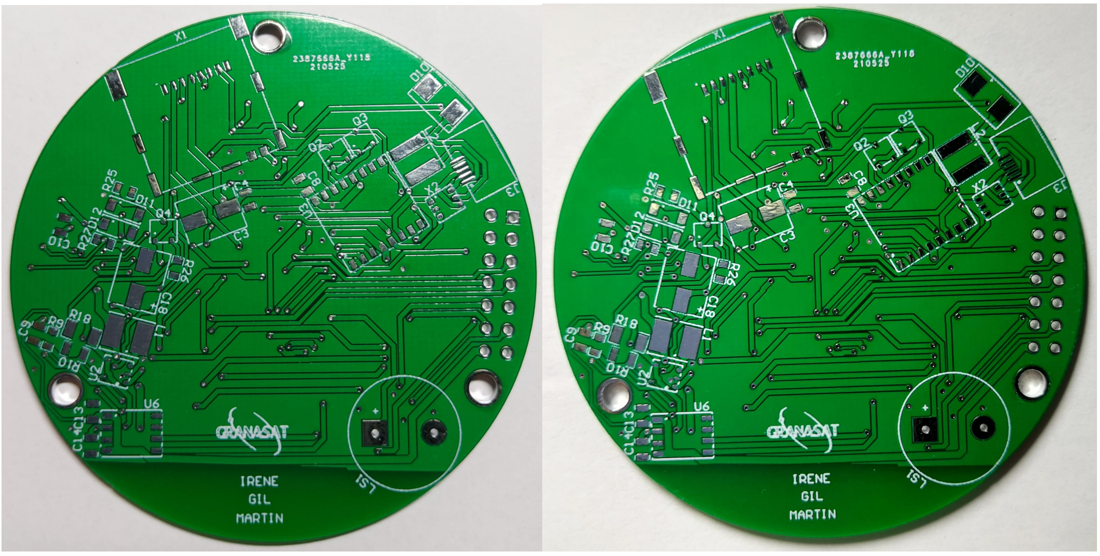
(a) *Top View* **(b)** *Bottom View*

Figure 5.1 – *GranaSAT CanSat Main Board PCB manufactured in the laboratory*

A video from the manufacturing is included

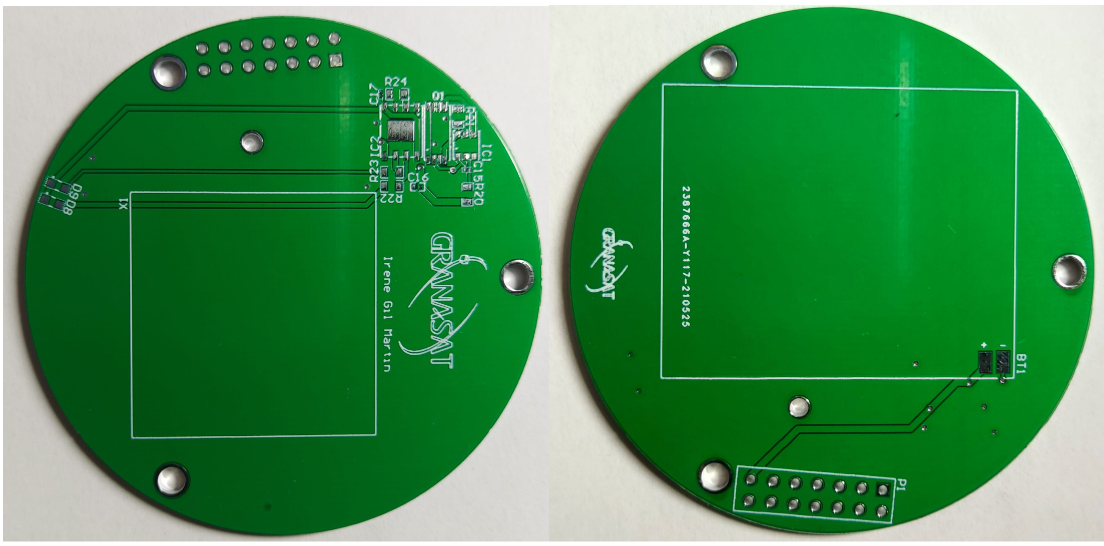
Video 5.1 – *Main Board manufacturing (double click in Adobe Acrobat)*

5.1.2 PCBs Industrially Manufactured



(a) *Top View* **(b)** *Bottom View*

Figure 5.2 – *GranaSAT CanSat Main Board PCB*

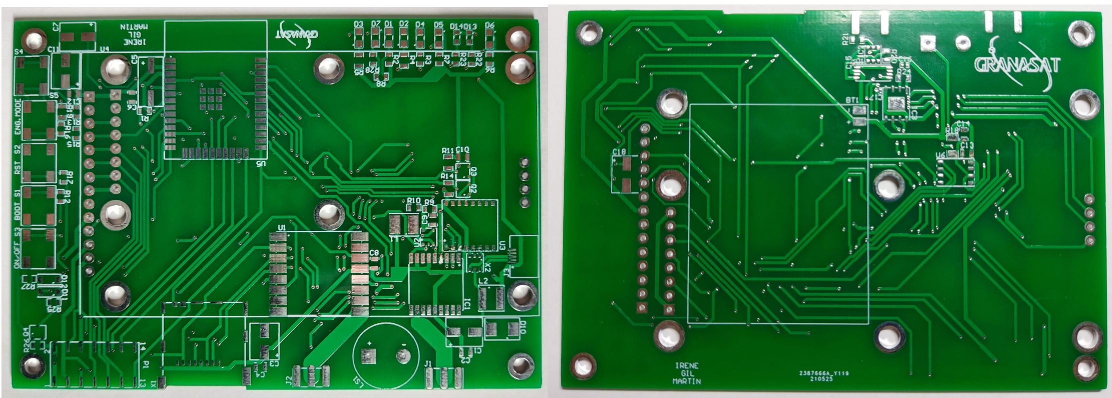


(a) *Top View* **(b)** *Bottom View*

Figure 5.3 – *GranaSAT CanSat Power Board PCB*

(a) *Top View* **(b)** *Bottom View*

Figure 5.4 – *GranaSAT CanSat Uni Board PCB*



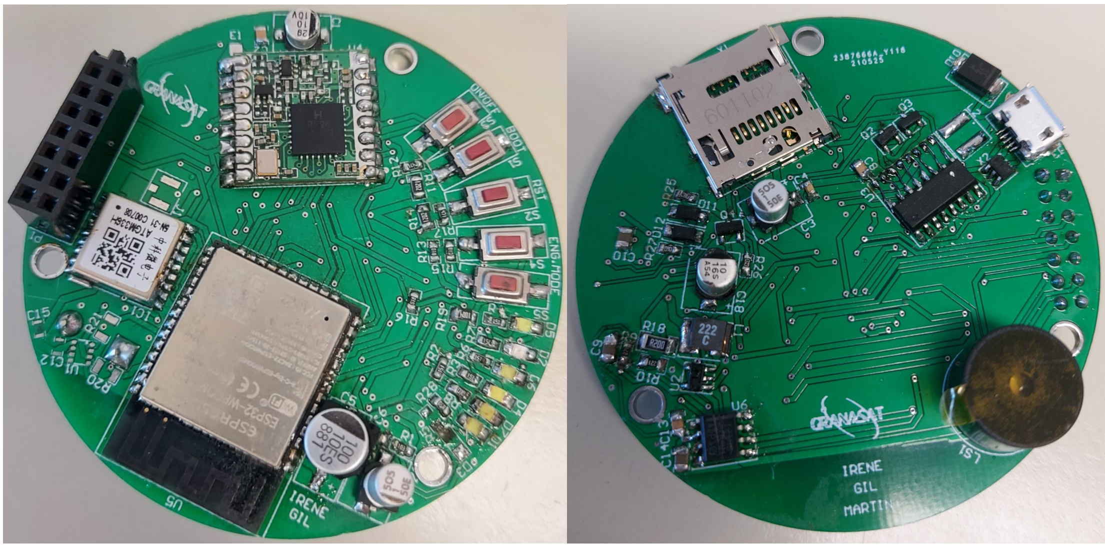
(a) *Top View* **(b)** *Bottom View*

Figure 5.5 – *GranaSAT CanSat Main Board PCB*

On the PCB manufacturing we only need to comment that when uploading the PCB files into the manufacturer web [46], we made the mistake of disabling the mass plane. Consequently, none of the the modules' have the GND connection. This aspect can be observe on the board transparency in Figure 5.5. The rest of the boards have been produced exactly as we intended.

5.1.3 PCBs soldered

We only soldered two boards: the Main Board and the Power Board. The Uni Board only has a LED, therefore as we run out of time, we decided not to lose any time in such a simple board. On the other hand, the Main Board does not have a mass plane, therefore the modules are not fully connected and will not work. We could have tried to connect them manually with wires, but as the circuits were most of them the same as the ones in the Main Board, we did not loose any time.



(a) *Top View* **(b)** *Bottom View*

Figure 5.6 – *GranaSAT CanSat Main Board PCB soldered*

While soldering this board we realized we made some mistakes with the modules footprint, for instance the CH340 has too much space between the pads. That's why we needed to solder some wires to make the connection. The cables we used were one-wire wires as the space is too small (5.6b).

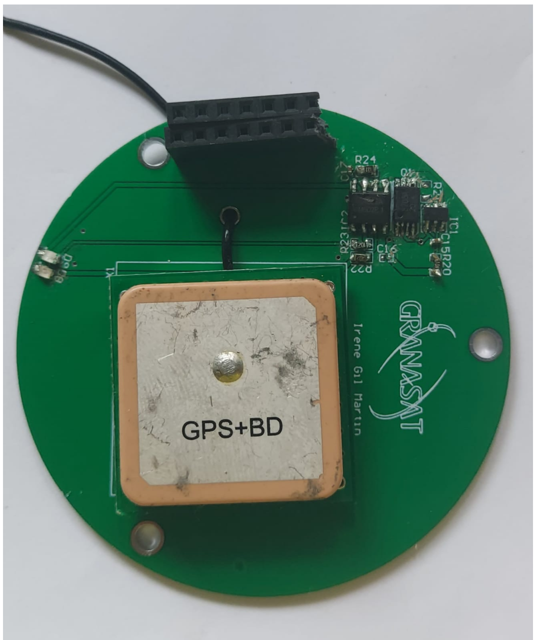


Figure 5.7 – *GranaSAT CanSat Main Board PCB soldered*

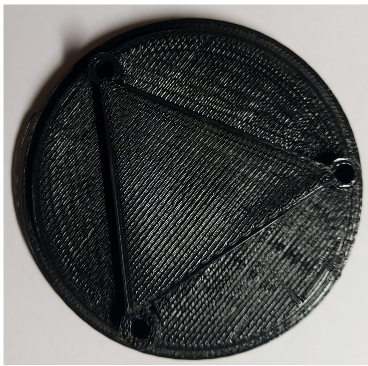
The only thing we needed to do with the Power Board was to desolder the TP4056 and solder it into this board. As we copied the circuit from the breakout board, this board is completely functional.

5.1.4 "Can" Assembly Production

In this section we will present the different images from the CanSat model.

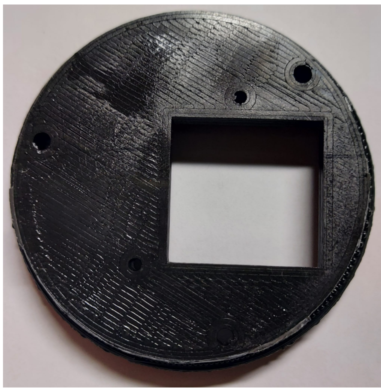
(a) *Side Perspective (I)*

(b) *Side Perspective (II)*

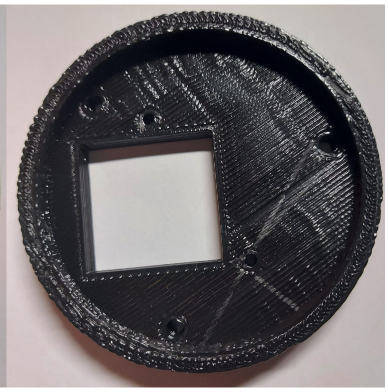


(c) *Top View*

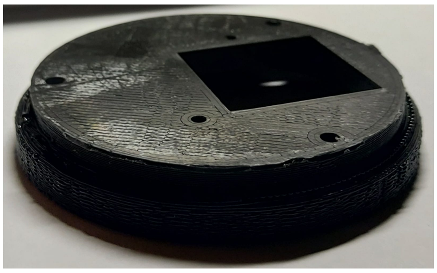
Figure 5.8 – *GranaSAT CanSat base "can"*



(a) *Top View*



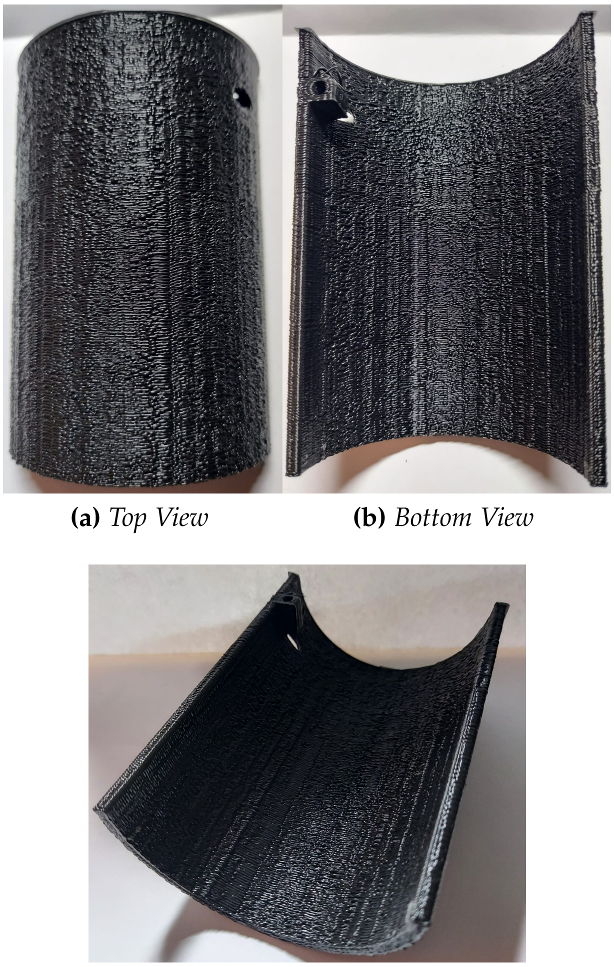
(b) *Bottom View*



(c) *Side Perspective*

Figure 5.9 – *GranaSAT CanSat cover "can"*

5



(a) *Top View* **(b)** *Bottom View*

(c) *Bottom View*

Figure 5.10 – *GranaSAT CanSat body "can" (left)*

(a) *Top View* **(b)** *Bottom View*

(c) *Bottom View*

Figure 5.11 – *GranaSAT CanSat cover "can"*

When printing this design we realized that the antenna hole was not well-positioned. Also we realized we could implement a more efficient model by using the spacers as hinges. In this new design, the different bodies would be assembled using screws that join the walls to the base, instead of simply joining the two walls.

A video of the process of printing this "can" can be found in 5.2.

Video 5.2 – *3D printer fabricating the "can" (double click in Adobe Acrobat)*

5.1.5 Circuits' Verifications

In this section we will be including different captures from the serial monitor and from the oscilloscope after executing the modules' software examples. This way we will be checking whether the circuits implemented are the right ones.

The order we will be following through this section is the one we followed when soldering the modules on the Main Board. Therefore we will start presenting the oscilloscope signals from the USB to serial conversion and move on to the rest of the modules one by one. The reason we solder the modules one by one checking its operation before soldering the rest of them was to avoid short-circuits.

Note that both boards, the Main Board and the Base Board, share the same circuits, therefore if we check whether they function on one of the board, the other board should function just the same way.

5.1.5.1 USB to serial conversion

First of all we soldered the circuits presented on page 12 of Subsection B.2.1 (page 12 of Subsection B.2.4) except the Reset/Booting microprocessor since the microprocessor was not solder yet. To check whether this three circuits were correctly implemented once they were soldered we only needed to connect the Main Board to the PC and check if the board was detected. It was in fact recognized as *Port USB-SERIAL CH*340*(COM*10*)*.

The last thing we needed to check before moving on to the next module was to make sure whether the programming circuit was correctly implemented. To so so, we prodded the oscilloscope on the *GPIO*0 and *RESET* pin and check their signals. The capture from the oscilloscope is presented in Figure 5.12.

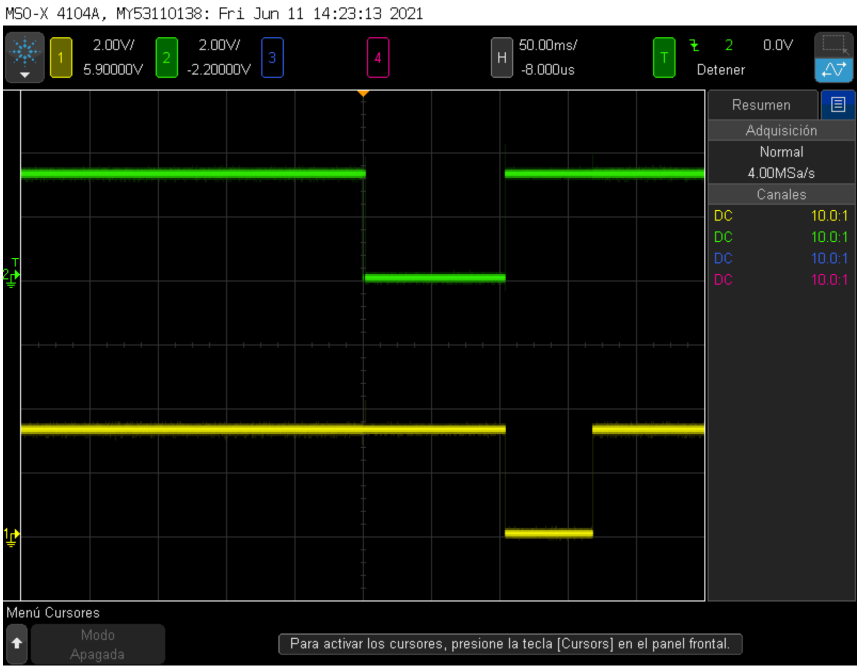


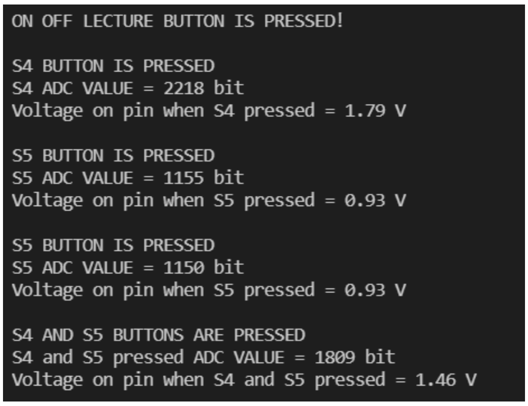
Figure 5.12 – *Programming pin signals in the oscilloscope*

We can observe that this signals are the same as the one we studied in Table 4.16. This signals (Figure 5.12) are the one needed for programming our microprocessor. Consequently, we can move on to solder this module.

5.1.5.2 Microprocessor and Analogue Buttons

Once we have checked we can communicate with the board and that we can program the microprocessor we moved on to solder the circuits needed for the glsuProcessor. This circuits are the one presented on page 6 of Subsection B.2.1 (same page on Subsection B.2.4). We also needed to solder the RESET/BOOTING circuits presented on page 12 of both Subsection B.2.1 and Subsection B.2.4.

Once this circuits were soldered, we needed to check whether the microprocessor could be programmed really. To test it, we uploaded the software to detect the analogue buttons described in paragraph 4.1.1.1.13. The capture from the serial monitor when uploading the test can be found in Figure 5.13.



```
ON OFF LECTURE BUTTON IS PRESSED!

S4 BUTTON IS PRESSED
S4 ADC VALUE = 2218 bit
Voltage on pin when S4 pressed = 1.79 V

S5 BUTTON IS PRESSED
S5 ADC VALUE = 1155 bit
Voltage on pin when S5 pressed = 0.93 V

S5 BUTTON IS PRESSED
S5 ADC VALUE = 1150 bit
Voltage on pin when S5 pressed = 0.93 V

S4 AND S5 BUTTONS ARE PRESSED
S4 and S5 pressed ADC VALUE = 1809 bit
Voltage on pin when S4 and S5 pressed = 1.46 V
```

Figure 5.13 – *Analogue Buttons detected*

Something we need to mention is that we made a mistake by confusing the microprocessor 4, 5, 6 and 7 pins as GPIO instead of GPI. Consequently, we needed to cut some of this tracks and change them with buttons tracks to make those pin work.

5.1.5.3 Power Switch

This captures correspond to the circuit we implemented for powering ON or OFF our device. This circuit was explained in paragraph 4.1.1.1.12. The button detection can be seen in the first line of Figure 5.13. However, the captures from the oscilloscope are the ones we are going to be interested since they are the ones that demonstrate the correct behavior we pretended to have.

In Figure 5.14 we can observe the signals related to the three main signals: +BAT signal (blue), ON_OFF_LECTURE signal (yellow) and the transistor drain signal (green). It can be seen that whenever the button is ON on the transistor is ON. However, when the button is not pressed, the transistor is OFF. This is just the behavior we looked for when designing the circuit.

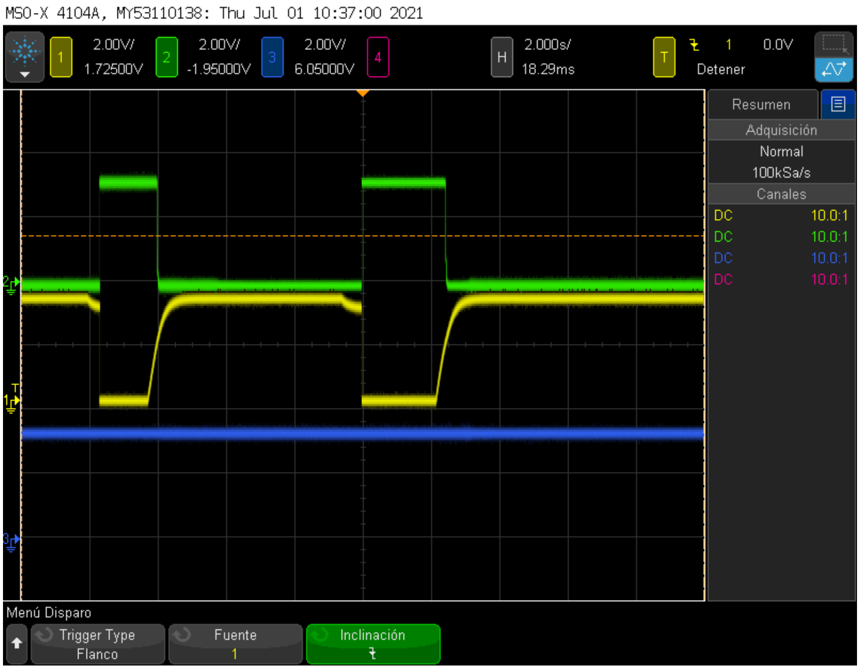
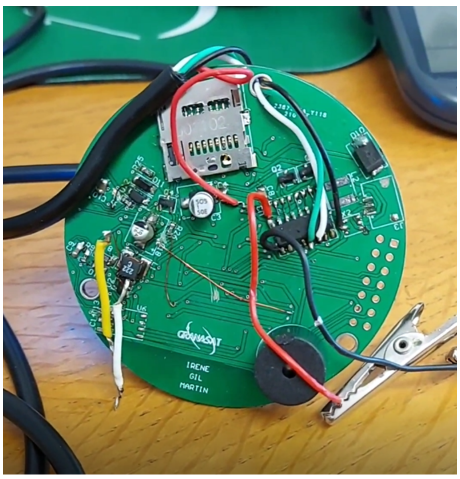


Figure 5.14 – *Power Switch oscilloscope signals*

5.1.5.4 Buzzer

Continuing to solder the modules, the next thing we included was the buzzer. The circuit implemented for this module is the one presented in 5.3.

The 5.3 shows (Adobe Reader or another compatible Video PDF reader needed) clip of the test corresponding to emitting a sound once each second.



Video 5.3 – *Buzzer test (double click in Adobe Acrobat)*

5.1.5.5 GPS Module

One of the last module we tested in this board is the GPS. We only needed to make sure that the strings were correctly received through the software UART we implemented in paragraph 4.1.1.1.5. Therefore we will only focus on receiving correctly the GPS strings and we will present them in a nice way later.

To better understand this communication protocol we podded the oscilloscope on both pins and observed the signals. This signals (TX (blue) and RX (yellow)) can be seen in Figure 5.15.

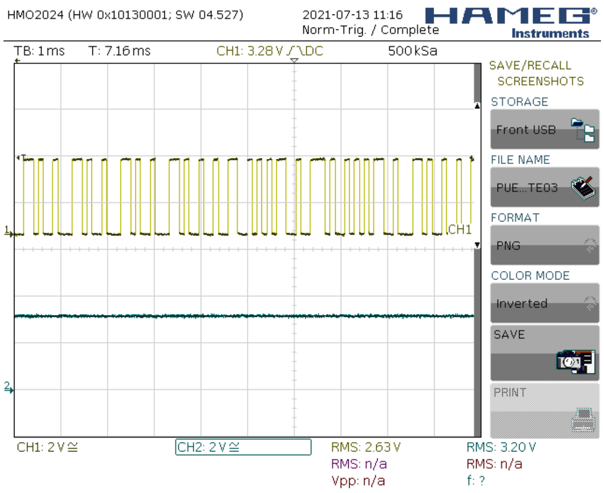


Figure 5.15 – *RS-232 signals from the GPS UART interface*

The strings received after the GPS synchronization are presented in Figure 5.16.

Figure 5.16 – *GPS received strings*

For reading this strings information we will refer to Section E.1. As we can observe from the *GPGGA* string, the data received is:

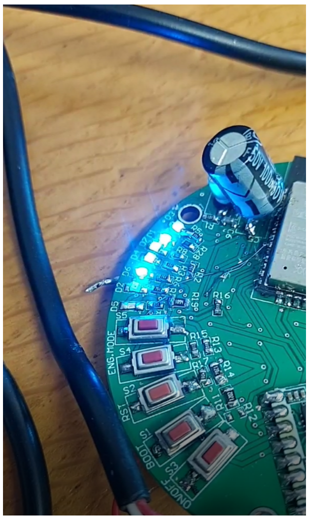
```
Time: 12:15:23.00
Latitude: 37º 11.34644' N
Longitude: 00º 336.32422' W
Fix Quality: GPS Fix
Number of Satellites: 6
Horizontal Dilution of Precision (HDOP): 7.6
Altitude: 714.6 m
Height of geoid above WGS84 ellipsoid: 0.
Time since last DGPS update: blank
DGPS reference station id: blank
Checksum: 2E
```

Figure 5.17 – *GPS GPGGA analyzed information*

5

5.1.5.6 LEDs

When testing the LEDs we discovered some of the main errors we committed. The first error we discovered was the confusion between the GPIO and GPI pins which affected some of this elements. Another issue we discover is that the double-implementation LED we described in Figure 4.13 was assigned to a non recommended SPI Flash pin. Even though we confirmed that the signals could be used in Figure 4.4, the LED requires much more current than the one this pin can proportionate, therefore it was impossible to program it. To test whether this configuration work as expect, we cut some tracks and assign this pin to another one. The results were favorable and 5.4 shows a clip (Adobe Reader or another compatible Video PDF reader needed) of the LEDs blinking.



Video 5.4 – *LEDs test (double click in Adobe Acrobat)*

5.1.5.7 Radio Module

As we did not plan on soldering the Base Board since it did not have any mass plane, we will only test whether this module can be programmed or not. We will not be testing the range or reliability of the communication.

When trying to program this module we realized that the HSPI pins we selected did not let us communicate with this protocol. Although this pin could be programmable as GPIOs. To check whether the radio module worked as planned, we decided to test in the NODE- we mentioned in previous sections. When connecting the module to the HSPI pins the result was the same, we could not talk to the module (SPI signals were flat). However, if we interfaced the same module to the VSPI pins, the module worked as expected. The capture for this module is as shown in Figure 5.19.

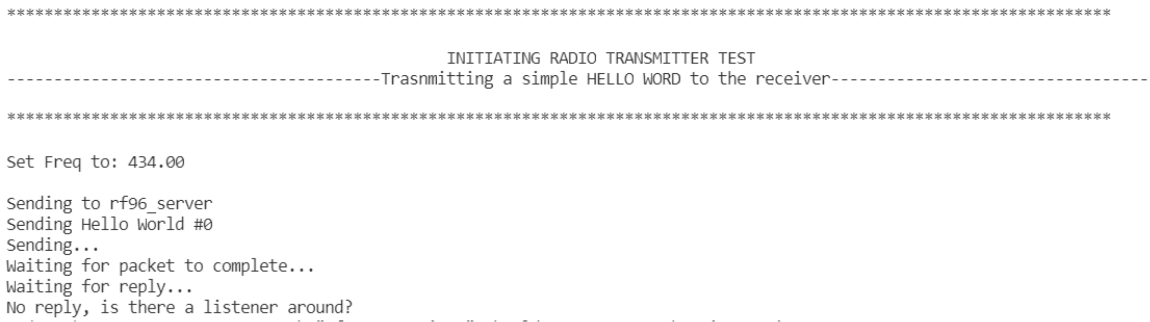


Figure 5.18 – *Radio Transmitter Test*

5.1.5.8 BME280 Sensor

We tested this module in the Node- given that we bought the breakout board instead of the sensor. Desoldering this module due to its size was a very difficult process we were unable to achieve.

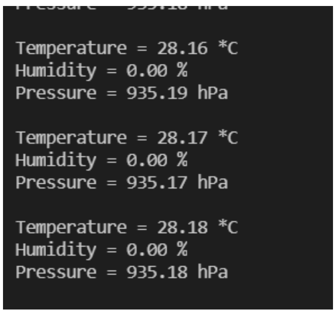


Figure 5.19 – *Radio Transmitter Test*

Note that on the humidity the value is 0.0 % due to the sensor implemented in the breakout board. This sensor is not the BME280 but is sibling sensor the BMP280.

5.1.5.9 INA219 Module

We did not get a chance to test this module since we committed a mistake with the footprint. This module measures the current and tension through it, therefore if we used some wires to correct the mistakes this values would be contaminated. Besides, as we already tested this module on the Czech engineers board we are not worried about the circuit functionality.

5.1.5.10 microSD connector and DC-DC buck converter

For testing the microSD connector we encounter one main problem. The soldering was too difficult since we bought a model with internal pins. Therefore reaching those pads with the solder without making any short-circuit was a difficult task. The last two modules we needed to test are the microSD and the buck converter. We encounter some problems to test this module.

5.1.5.11 DC-DC buck converter

The last module we needed to test on this board is the buck converter. Even though his footprint and connections were correctly made, the INA219 footprint error contaminated this module signals and we were able to make it work.

5.1.5.12 LCD module

The last module we needed to check was the LCD. Since we did not solder the Base Board we tested this module on the Node-. When trying to interface this module with the HSPI pins we experimented some errors, therefore we decided to interface the module through the VSPI pins. Even though we could program the display, we were not able to use both the display and the touch sensor although we changed the CS pin. Therefore we will need to try to interface the LCD with the I2C interface and use the VSPI interface for the touch sensor. Since we run out of time, that something we have left pending.

Figure 5.20 – *LCD Test*

With this section we conclude the circuits' verifications.

5

Chapter 6

Conclusions and Future Lines

This document has shown the tough procedure of performing a reverse engineering process to an existing product. To do such thing, we have presented the development of a CanSat from prototype to final design through the different stages that entails. We have managed to understand and familiarized with the process of a system design in the engineering industry. This Bachelor's Thesis has allowed a telecommunication engineering student, as the author, to better understand the concepts learned during the academic years of electronic design and radio communication.

During the development of this work, especially at the beginning of the project, we realized that the student's base knowledge was not enough to tackle this project. Although he knew the basic notions learned during the career, the lack of experience made the first phases of the project slower. In the course of the academic period, there has not been so much emphasis on bringing the student closer to what the industry really is as much as the theoretical knowledge, therefore, the student does not know well how to function when designing a product or even getting along at the laboratory

Throughout this time period we have learned a little about how this industry works, its standards and how important a good documentation is. We have also learn how to work in a science lab with other partners, where everyone can learn from each other even though the project's they are working in are different.

The CanSat developed is yet far from being implemented and ready to be sold since we only did the first prototype. However, we think we have lain the groundwork for this purpose. Besides, we have proven that the hardware implemented to be quite robust. As for the software, even though we have set a very solid ground on this aspects, we will need to correct several hardware aspects in order to be able to develop the device's complete firmware.

The lines we need to work on in the future to get this product ready to be sold are:

- Microprocessor pin reassignment
- Correct erroneous footprints
- Uploading the correct PCBs files into the manufacturer page (solving the missing mass plane on the Base Board)
- Redesigning a better ”can” for the transmitter using the spacers as hinges for a more robust design
- Fixing the errors committed with the PCBs design

Even though we have set a quite basic kid, in future lines we could work on developing a much more ambitious design able to perfecting the flight part. We could be focusing on adding some *NEMA* motors and some helixes for trying to fixate the landing.

We would like to conclude by saying that it has been quite an enriching experience developing this project, and we did not expect us to have this ability to adapt to the problems and difficulties encountered along a project of this type.

Once again, the future is exciting and we hope to work on perfect this design until it is ready to be sold for the students use.

References

[1] About cubesat. https://alen.space/es/guia-basica-nanosatelites/.

[2] Britannica, the editors of encyclopaedia. november-2019. *"Microprocessor"*.accessed 6 september 2021. https://www.britannica.com/technology/microprocessor.

[3] Czech engineers, (2019). *Open CanSat GitLab*, (13-07-2021). http://gitlab.sciencein.cz/esero_public_projects/opencansat.

[4] Esero (european office of resources for space education in spain) web. http://esero.es/.

[5] Esero requirements for 2020/2021 competition. https://esero.es/wp-content/uploads/2021/06/basescansat2021-actualizada0406.pdf.

[6] European Space Agency. https://www.esa.int/ESA.

[7] Goñi,jose antonio (march, 2021). *CanSat* 2021 *en el Circuito de Navarra*. accessed 12 july 2021. https://www.diariodenavarra.es/multimedia/galerias-imagenes/navarra/2021/03/21/cansat-2021-el-circuito-navarra.html.

[8] Granasat.ugr.es, *Granasat Offcial Website*. [online]. https://granasat.ugr.es/.

[9] Hommer. *Eagle VS Altium-What Is The Features And Differences*. accessed 6 august 2021. https://www.wellpcb.com/eagle-vs-altium.html.

[10] Npd-solutions, (2015,aug). *A methodology for Reverse Engineering*. accessed 12 june 2021. https://www.npd-solutions.com/remethodology.html.

[11] Open cansat project web from a czech republic team. http://kit.sciencein.cz/#.

[12] *Bluetooth Module*, vlaserelectronica.com [online]. https://vlaserelectronica.com/producto/hc-08-bluetooth-ble-4-0/. Accessed 7 June 2021.

[13] *Patch Antenna*, tme.eu [online]. https://www.tme.eu/es/details/org12-4t-gnss/antenas-gps/origingps/. Accessed 4 June 2021.

[14] *Li-Polymer Charging Module datasheet*, microchip.com [online]., 2005. https://ww1.microchip.com/downloads/en/DeviceDoc/MCP73831-Family-Data-Sheet-DS20001984H.pdf. Accessed 11 June 2021.

[15] *KXG1205 Buzzer Datasheet*, pdf1.alldatasheet.com [online], 2006. https://pdf1.alldatasheet.com/datasheet-pdf/view/512368/ETC1/KXG1205.html. Accessed 6 August 2021.

[16] *RFM69 ISM TRANSCEIVER MODULE Datasheet*, hoperf.com [online]., 2006. https://www.hoperf.com/modules/rf_transceiver/RFM69W.html?/modules/index.html&gclid=Cj0KCQjwnueFBhChARIsAPu3YkRh0nrDCfH0x08pj0UoxY9C6c4MHYvb_C0nT7HMfpLIhob3Ffd60vQaAiwcEALw_wcB. Accessed 4 June 2021.

[17] *LDO regulator*, microchip.com [online]., 2007. https://ww1.microchip.com/downloads/en/DeviceDoc/22057B.pdf. Accessed 11 June 2021.

[18] *LDO regulator*, octopart.com [online]., 2007. https://duino4projects.com/mcp73831-youll-charge-lithium-cells-easily-safely/. Accessed 11 June 2021.

[19] *Li-Polymer Charging Module*, duino4projects.comm [online]., 2007. https://octopart.com/adp3338akcz-5-reel-analog+devices-527682. Accessed 11 June 2021.

[20] *Display ST7735 datasheet*, datasheetpdf.com [online], 2008. https://datasheetspdf.com/pdf-file/694957/SitronixTechnology/ST7735/1. 20 June 2021.

[21] *BME280 Sensor Datasheet*, bosch-sensortec.com [online]., 2012. https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf. Accessed 4 June 2021.

[22] *BMP280 Sensor Datasheet*, bosch-sensortec.com [online]., 2012. https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp280-ds001.pdf. Accessed 4 June 2021.

[23] *ESD protector datasheet*, digikey.es [online]., 2012. https://www.st.com/content/ccc/resource/technical/document/datasheet/group0/94/15/6c/58/ba/91/46/42/DM00055262/files/DM00055262.pdf/jcr:content/translations/en.DM00055262.pdf. Accessed 11 June 2021.

[24] *Bluetooth Module Datasheet*, electronicoscaldas.com [online]., 2014. https://www.electronicoscaldas.com/datasheet/HC-Serial-Bluetooth-Products-User-Instructional-Manual_Wavesen.pdf. Accessed 7 June 2021.

[25] *INA219 image*, ti.com [online]., 2014. https://www.ti.com/store/ti/en/p/product/?p=INA219BIDCNT. Accessed 7 June 2021.

[26] *Patch Antenna Datasheet*, taoglass.com [online]., 2014. https://www.taoglas.com/images/product_images/original_images/Internal%20GPS%20Active%20Patch%20Antenna(APN-13-8-002.A).pdf. Accessed 4 June 2021.

[27] *Radio Module Library*, github.com [online], 2014. https://github.com/PaulStoffregen/RadioHead. Accessed 10 July 2021.

[28] *SAMD21 microprocessor Family Datasheet*, microchip.com [online]., 2014. https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-D21DA1-Family-Data-Sheet-DS40001882G.pdf. Accessed 4 June 2021.

[29] *SAMD21 microprocessor*, tme.com [online]., 2014. https://www.tme.com/mx/es/details/samd21g18a-aut/familia-arm-32-bit/microchip-atmel/atsamd21g18a-aut/. Accessed 4 June 2021.

[30] 433 *MHz Antenna information*, es.aliexpress.com [online]., 2015. https://es.aliexpress.com/item/33005665407.html?spm=a2g0o.productlist.0.0.270b77790Wm4xV&algo_pvid=51f4fa97-8edf-4d62-aaf1-d959430970fd&algo_expid=51f4fa97-8edf-4d62-aaf1-d959430970fd-0&btsid=2100bde716135805803467450ef7b3&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_. Accessed 3 August 2021.

[31] *AP3429/A buck converter datasheet*, datasheetpdf.com [online], 2015. https://datasheetspdf.com/pdf-file/974154/Diodes/AP3429/1. Accessed 18 July 2021.

[32] *ATGM336H datasheet*, icofchina.com [online], 2015. http://www.icofchina.com/d/file/xiazai/2016-12-05/b5c57074f4b1fcc62ba8c7868548d18a.pdf. Accessed 16 July 2021.

[33] *ATGM336H Image*, doitsmart.hu [online], 2015. https://www.doitsmart.hu/wp-content/uploads/2020/11/ATGM336H-5N-31.jpg. Accessed 18 July 2021.

[34] *Battery information*, es.aliexpress.com [online], 2015. https://es.aliexpress.com/item/1005001307809914.html?spm=a2g0o.productlist.0.0.656b4418m8Aexq&algo_pvid=e45b07ce-37af-4777-9e1f-636b0bd9c080&algo_expid=e45b07ce-37af-4777-9e1f-636b0bd9c080-53&btsid=0b0a0ac216209198811644542ee169&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_. Accessed 25 July 2021.

[35] *Bootloader definition*, students.iitk.ac.in[online]., 2015. https://students.iitk.ac.in/eclub/assets/tutorials/Bootloader.pdf. Accessed 20 August 2021.

[36] *CH340 module datasheet*, alldatasheet.com [online], 2015. https://pdf1.alldatasheet.com/datasheet-pdf/view/1132602/WCH/CH340.html. Accessed 18 July 2021.

[37] *DW01A datasheet*, best-microcontroller-projects.com [online], 2015. https://www.best-microcontroller-projects.com/support-files/dw01a.pdf. Accessed 25 July 2021.

[38] *ESP-32-WROOM-32D 3D model*, grabcad.com [online]., 2015. https://grabcad.com/library/esp-wroom-32-1. Accessed 3 August 2021.

[39] *FS8205A datasheet*, pdf1.alldatasheet.com [online], 2015. https://datasheet.lcsc.com/szlcsc/2010271837_FUXINSEMI-FS8205A_C908265.pdf. Accessed 25 July 2021.

[40] *GPS Antenna information*, es.aliexpress.com [online]., 2015. https://es.aliexpress.com/item/1005001970214642.html?spm=a2g0o.productlist.0.0.1e1538231QU5cb&algo_pvid=370ddf4c-f724-4da2-84bf-c1118d96b48d&algo_expid=370ddf4c-f724-4da2-84bf-c1118d96b48d-1&btsid=2100bdcf16140351945351936ebc24&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_. Accessed 3 August 2021.

[41] *HDMIULC6-4SC6 ESD protector*, verical.com [online], 2015. https://www.verical.com/datasheet/stmicroelectronics-esd-supressors-HDMIULC6-4SC6-309985.pdf. Accessed 18 July 2021.

[42] *ILI9341 information*, es.aliexpress.com [online], 2015. https://es.aliexpress.com/item/1005001626547564.html?spm=a2g0o.productlist.0.0.35f613c0eWBt15&algo_pvid=238cb6ba-1085-462e-9133-bac55f0bc133&algo_expid=238cb6ba-1085-462e-9133-bac55f0bc133-31&btsid=2100bdf016139901318562821eca19&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_. Accessed 3 August 2021.

[43] *In-Circuit Debuggers and SWD/JTAG Standard*, embeddedinventor.com [online]., 2015. https://embeddedinventor.com/swd-vs-jtag-differences-explained/. Accessed 20 August 2021.

[44] *In-Circuit Debuggers Definition*, sciencedirect.com [online]., 2015. https://www.sciencedirect.com/topics/engineering/circuit-debugger. Accessed 20 August 2021.

[45] *INA219 datasheet*, ti.com [online]., 2015. https://www.ti.com/lit/ds/symlink/ina219.pdf?ts=1625946854537&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FINA219. Accessed 7 June 2021.

[46] *JLPCB PCB manufacturer*, jlcpcb.com [online]., 2015. https://jlcpcb.com/. Accessed 3 August 2021.

[47] *Logic Analyzer*, amazon.es [online]., 2015. https://www.amazon.es/AZDelivery-Logic-Analyzer-compatible-versi%C3%B3n/dp/

B01MUFRHQ2/ref=sr_1_1_sspa?adgrpid=73954943444&dchild=1&gclid=
CjwKCAjw64eJBhAGEiwABr9o2BVcrEZ63yR0USe11Gr0jqe0XwutWFGwdepp3c31N4GiHktHhuQZrBoCjt
BwE&hvadid=357903904739&hvdev=c&hvlocphy=9047061&hvnetw=g&
hvqmt=b&hvrand=13540457410475654773&hvtargid=kwd-421219791568&
hydadcr=13808_1743183&keywords=azdelivery+logic+analyzer&
qid=1629676514&sr=8-1-spons&psc=1&smid=A1X7QLRQH87QA3&spLa=
ZW5jcnlwdGVkUXVhbGlmaWVyPUEzS1RMVlBSRElDU1BUJmVuY3J5cHRlZElkPUEwNTAwOTQzM0QySFE4SE
Accessed 17 August 2021.

[48] *microUSB Type-B*, trgroupk.com [online]., 2015. https://www.trgroupk.com/index.php?main_page=product_info&products_id=267483. Accessed 7 June 2021.

[49] *MOLEX 473460001 USB*, molex.com [online], 2015. https://www.molex.com/webdocs/datasheets/pdf/en-us/0473460001_IO_CONNECTORS.pdf. Accessed 18 July 2021.

[50] *NMEA data strings*, gpsworld.com[online]., 2015. https://www.gpsworld.com/what-exactly-is-gps-nmea-data/. Accessed 14 August 2021.

[51] *NMEA data strings header classification*, trimble.com[online]., 2015. https://www.trimble.com/oem_receiverhelp/v4.44/en/NMEA-0183messages_MessageOverview.html. Accessed 14 August 2021.

[52] *NMEA data strings information*, rfwireless-world.com[online]., 2015. https://www.rfwireless-world.com/Terminology/GPS-sentences-or-NMEA-sentences.html. Accessed 14 August 2021.

[53] *RTOS introduction*, digikey.com [online]., 2015. https://www.digikey.com/en/maker/videos/shawn-hymel/getting-started-with-stm32-and-nucleo-part-3-how-to-run-
Accessed 24 August 2021.

[54] *SEGGER official page*, segger.com[online]., 2015. https://www.segger.com/. Accessed 20 August 2021.

[55] *SysProgs USB Driver Tool download link*, visualgdb.com [online]., 2015. https://visualgdb.com/UsbDriverTool/. Accessed 20 August 2021.

[56] *TP4056 battery charger datasheet*, drive.google.com [online], 2015. https://drive.google.com/file/d/12REE5_bEsn3-qgz9wvhcnFtNy99jGvwZ/view. Accessed 18 July 2021.

[57] *TP4056 chip's datasheet*, dlnmh9ip6v2uc.cloudfront.net [online], 2015. https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf. Accessed 25 July 2021.

[58] *TP4056 chip's information*, best-microcontroller-projects.com [online], 2015. https://www.best-microcontroller-projects.com/tp4056.html. Accessed 25 July 2021.

[59] *VNA manual page*, ttrf.tk[online]., 2015. https://ttrf.tk/kit/nanovna/. Accessed 17 August 2021.

[60] *GPS Module Library*, github.com [online], 2016. https://github.com/mikalhart/TinyGPSPlus. Accessed 10 July 2021.

[61] *GPS Receiver Datasheet*, u-blox.com [online]., 2016. https://www.u-blox.com/sites/default/files/MAX-8_DataSheet_%28UBX-16000093%29.pdf. Accessed 4 June 2021.

[62] *LCD Module Library*, github.com [online], 2017. https://github.com/olikraus/ucglib. Accessed 10 July 2021.

[63] *MicroSD Card Connector Datasheet*, molex.com [online]., 2018. https://www.molex.com/molex/products/part-detail/memory_card_socket/5027740891, Accessed 4 June 2021.

[64] *BME280 Sensor Library*, github.com [online], 2019. https://github.com/adafruit/Adafruit_BME280_Library. Accessed 10 July 2021.

[65] *ESP32 datasheet*, espressif.com [online], 2019. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. Accessed 16 July 2021.

[66] *ESP32 family information*, en.wikipedia.org [online], 2019. https://en.wikipedia.org/wiki/ESP32#ESP32-xx_family. Accessed 16 July 2021.

[67] *ESP32 restoring bootloader*, mcuoneclipse.com [online], 2019. https://mcuoneclipse.com/2019/09/22/eclipse-jtag-debugging-the-esp32-with-a-segger-j-link/. Accessed 16 July 2021.

[68] *INA219 Module Library*, github.com [online], 2019. https://github.com/adafruit/Adafruit_INA219. Accessed 10 July 2021.

[69] *microSD datasheet*, molex.com [online], 2019. https://www.molex.com/molex/products/part-detail/memory_card_socket/5045280892. Accessed 16 July 2021.

[70] *NODEMCU ESP32S schematic*, smart-prototyping.com [online], 2019. https://www.smart-prototyping.com/image/data/2_components/development_board/101839%20NodMCU-32S/1.png. Accessed 16 July 2021.

[71] *RFM96W datasheet*, hoperf.com [online], 2019. https://www.hoperf.com/modules/lora/RFM96.html. Accessed 16 July 2021.

[72] *SD Module Library*, github.com [online], 2019. https://github.com/arduino-libraries/SD. Accessed 10 July 2021.

[73] *Software RTC Library*, github.com [online], 2019. https://github.com/PaulStoffregen/Time. Accessed 10 July 2021.

[74] *Interfacing J-Linker with the board with the SWD interface*, segger.com [online], 2020. https://www.segger.com/products/debug-probes/j-link/technology/interface-description/#swd-and-swo-also-called-swv-compatibility. Accessed 17 June 2021.

[75] *Super Loop Architecture*. en.wikibooks.org [online], 22-5-2019. https://en.wikibooks.org/wiki/Embedded_Systems/Super_Loop_Architecture. Accessed 20 June 2021.

[76] Bolanos, R. *How to model a Diode using a Datasheet*, 06-01-2020. https://www.youtube.com/watch?v=nP5alFwYBH8. Accessed 6 June 2021.

[77] Challa, V. *How To Select The Right Battery For Your Application? Part 1: Important Battery Metric Considerations*, dfrsolutions.com [online], 20-01-2017. https://www.dfrsolutions.com/blog/how-to-select-the-right-battery-for-your-application-part-1-battery-metric-consid Accessed 25 July 2021.

[78] Farahani, S. *RF Propagation, Antennas, and Regulatory Requirements.* sciencedirect.com [online], 2008. https://www.sciencedirect.com/science/article/pii/B9780750683937000054. Accessed 4 June 2021.

[79] Kumar, A. *Difference between I2C and SPI (I2C vs SPI), you should know.* articleworld [online], 2018. https://aticleworld.com/difference-between-i2c-and-spi/. Accessed 4 June 2021.

Appendix A

A Project Budget

A.1 Materials and hardware

In this section, project cost regarding materials and hardware sections will be detailed. Each one of the different hardware subsections is differentiated. Human resources area also included.

A.1.1 GranaSAT CanSat Device

The cost of the device will include the electronic components, the cost of the PCBs manufacturing and also the cost of the PLA material for the "can".

A.1.1.1 GranaSAT CanSat Can

The "can" has been fabricated with a 3D printer. The material used for this printer is PLA which has a cost of $0,12$ €/g. Given that in Subsection 4.1.4 we saw that the final weight of this "can" was 56 g, the total cost would be 6.72 €.

CanSat Can Total Cost	**6.72 €**

Table A.1 – *PLA material total cost*

A.1.1.2 CanSat Electronic Modules

On this section we will be presenting the total cost of the electronic devices we have used for the four boards' circuits implementation. Since our device will be divided into a transmitter and into a receiver, we will be presenting the cost with that division.

A.1.1.2.1 GranaSAT CanSat Transmitter

Item	Module	Units	Cost/u. (€)	Cost (€)
ESP32-WROOM-32D	Microprocessor	1	2.20 €	2.20 €
BME280	T,P,H Sensor	1	1.21 €	1.21 €
MOLEX504528	MicroSD	1	2.19 €	2.19 €
RFM96	Radio	1	3.26 €	3.26 €
ATGM336H	GPS	1	2.77 €	2.77 €
Ceramic Antenna	Patch Antenna	1	2.84 €	2.84 €
IPX Connector	Antenna Connector	1	0.67 €	0.67 €
SMD 0805	LEDs	7	0.15 €	1.06 €
KXG1205	Buzzer	1	0.47 €	0.47 €
INA219	Current Monitor	1	0.77 €	0.77 €
MOLEX47346	USB	1	0.83 €	0.83 €
HDMIULC6	ESD PROTECTOR	1	0.55 €	0.55 €
CH340C	USB-Serial	1	0.68 €	0.68 €
2N7002	NPN MOSFET	2	0.12 €	0.25 €
AP3429/A	Buck Converter	1	0.34 €	0.34 €
Pin Header	Bus Connector	3	1.41 €	4.23 €
BSS84	PNP MOSFET	1	0.19 €	0.19 €
1N4148	Diode	2	0.11 €	0.23 €
DO-214AAA	Diode	1	0.28 €	0.28 €
404050 Li-On	Battery	1	5.10 €	5.10 €
TP4056	Battery Charger	1	0.47 €	0.47 €
SMD Buttons	Buttons	5	0.51 €	2.55 €
SMD Ind	Inductor	2	0.22 €	0.44 €
SMD Res	Resistor	22	0.08 €	1.87 €
SMD Cap	Capacitor	7	0.08 €	0.59 €
Electrolytic Cap	Capacitor	4	0.13 €	0.52 €
SubTotal (before VAT): 28.50 €				
Total (VAT included): 36.08 €				

Table A.2 – *GranaSAT CanSat transmitter's cost*

A.1.1.3 GranaSAT CanSat Receiver

Item	Module	Units	Cost/u. (€)	Cost (€)
ESP32-WROOM-32D	Microprocessor	1	2.20 €	2.20 €
ILI9341	LCD	1	12.63 €	12.63 €
MOLEX504528	MicroSD	1	2.19 €	2.19 €
RFM96	Radio	1	3.26 €	3.26 €
433 MHz antenna	Patch Antenna	1	5.22 €	5.22 €
ATGM336H	GPS	1	2.77 €	2.77 €
SMA antenna	Lorawan Antenna	1	3.12 €	3.12 €
Female SMA	Antenna Connector	2	1.74€	3.48 €
SMD 0805	LEDs	7	0.15 €	1.06 €
KXG1205	Buzzer	1	0.47 €	0.47 €
INA219	Current Monitor	1	0.77 €	0.77 €
MOLEX47346	USB	1	0.83 €	0.83 €
HDMIULC6	ESD PROTECTOR	1	0.55 €	0.55 €
CH340C	USB-Serial	1	0.68 €	0.68 €
2N7002	NPN MOSFET	2	0.12 €	0.25 €
AP3429/A	Buck Converter	1	0.34 €	0.34 €
Pin Header	Bus Connector	3	1.41 €	4.23 €
BSS84	PNP MOSFET	1	0.19 €	0.19 €
1N4148	Diode	2	0.11 €	0.23 €
DO-214AAA	Diode	1	0.28 €	0.28 €
404050 Li-On	Battery	1	5.10 €	5.10 €
TP4056	Battery Charger	1	0.47 €	0.47 €
SMD Buttons	Buttons	5	0.51 €	2.55 €
SMD Ind	Inductor	2	0.22 €	0.44 €
SMD Res	Resistor	22	0.08 €	1.87 €
SMD Cap	Capacitor	8	0.08 €	0.64 €
Electrolytic Cap	Capacitor	5	0.13 €	0.65 €
SubTotal (before VAT): 44.61 €				
Total (VAT included): 56.47 €				

Table A.3 – *GranaSAT CanSat transmitter's cost*

A.1.2 PCBs Manufacturing

The manufacturing of a 2-layers PCBs in *https://jlcpcb.com/* has a cost of 3.27 €[1]. Note that as we had an offer where one of the PCBs came out for half the price. The cost of a stencil, in which we were able to fit the top layer of the four boards, is 5.77 €.

Product Manufactured	Cost
PCBs Manufacturing Cost (without shipping cost)	**17.19 €**
Stencil Manufacturing Cost (without shipping cost)	**5.77 €**
PCBs and Stencil Manufacturing Total Cost (with shipping cost)	**33.85 €**

Table A.4 – *PCBs and stencil total cost*

A.2 Open CanSat vs GRANASAT cost

The last thing we need to do on this project budget is trading-off whether our product is more economical than the one the Czech engineers designed or not. To do so, we will be comparing the electronic device cost of each product. This will give us the perfect information of which product will be more economic.

A.2.1 Open CanSat electronic cost

As we do not have their porject's budget, we will be searching for the price on the Internet. Therefore, this may not be the most accurate information. To be able to compare both project's costs we will be taking the passive elements and the LEDs as if the cost was the same for both of them.

A.2.1.1 Open CanSat transmitter

Item	Module	Units	Cost/u. (€)	Cost (€)
SAMD21G18A-AUT	Microprocessor	1	2.73 €	2.73 €
BME280	T,P,H Sensor	1	1.21 €	1.21 €
MOLEX346532	MicroSD	1	2.77 €	2.77 €
RFM69	Radio	1	1.52 €	1.52 €
MAX-M8	GPS	1	5.62 €	5.62 €
Ceramic Antenna	Patch Antenna	1	3.10 €	3.10 €
SMD 0805	LEDs	8	0.15 €	1.20 €
INA219	Current Monitor	1	0.77 €	0.77 €
10118193	USB	1	0.26 €	0.26 €

Continued on next page

[1]This is the price for 5 PCBs of each board since they do not let you manufacture only one.

Item	Module	Units	Cost/u. (€)	Cost (€)
USBLC6	ESD PROTECTOR	1	0.36 €	0.36 €
MCP7381T	Battery Charger	1	0.48 €	0.68 €
IRLM6346	NPN MOSFET	1	0.20 €	0.20 €
MCP1826S	Buck Converter	1	0.83 €	0.83 €
Pin Header	Bus Connector	3	1.41 €	4.23 €
32.768 kHz Xtal	Crystal Oscillator	1	0.22 €	0.22 €
SMAJ6.0CA	Diode	2	0.48 €	0.96 €
500 mA PTC	Fuse	2	0.58 €	1.16 €
404050 Li-On	Battery	1	3.33 €	3.33 €
SMD Buttons	Buttons	2	0.51 €	1.02 €
SMD Ind	Inductor	2	0.22 €	0.44 €
SMD Res	Resistor	17	0.08 €	1.36 €
SMD Cap	Capacitor	23	0.08 €	1.84 €
Electrolytic Cap	Capacitor	6	0.13 €	0.78 €
SubTotal (before VAT): 28.90 €				
Total (VAT included): 36.59 €				

Table A.5 – *Open CanSat transmitter's cost*

A.2.1.2 Open CanSat receiver

Item	Module	Units	Cost/u. (€)	Cost (€)
SAMD21G18A-AUT	Microprocessor	1	2.73 €	2.73 €
TFT 1.8″	LCD	1	3.88 €	3.88 €
MOLEX346532	MicroSD	1	2.77 €	2.77 €
RFM69	Radio	1	1.52 €	1.52 €
MAX-M8	GPS	1	5.62 €	5.62 €
Female SMA	Antenna Connector	2	1.74€	3.48 €
SMD 0805	LEDs	8	0.15 €	1.20 €
ZS-040	Bluetooth	1	6.95 €	6.95 €
10118193	USB	1	0.26 €	0.26 €
USBLC6	ESD PROTECTOR	1	0.36 €	0.36 €
MCP7381T	Battery Charger	1	0.48 €	0.68 €
IRLM6346	NPN MOSFET	1	0.20 €	0.20 €
MCP1826S	Buck Converter	1	0.83 €	0.83 €
Pin Header	Bus Connector	3	1.41 €	4.23 €
32.768 kHz Xtal	Crystal Oscillator	1	0.22 €	0.22 €

Continued on next page

Item	Module	Units	Cost/u. (€)	Cost (€)
SMAJ6.0CA	Diode	2	0.48 €	0.96 €
500 mA PTC	Fuse	2	0.58 €	1.16 €
404050 Li-On	Battery	1	3.33 €	3.33 €
SMD Buttons	Buttons	5	0.51 €	2.55 €
SMD Ind	Inductor	2	0.22 €	0.44 €
SMD Res	Resistor	17	0.08 €	1.36 €
SMD Cap	Capacitor	23	0.08 €	1.84 €
Electrolytic Cap	Capacitor	6	0.13 €	0.78 €
SubTotal (before VAT): 37.30 €				
Total (VAT included): 47.35 €				

Table A.6 – *Open CanSat receiver's cost*

A.2.2 Product's trade-off

Given that we need the final cost of both transmitter and receiver, we will be comparing its prices. The price's comparative is exposed in Table A.2.2.

Open CanSat Total Electronic Cost	**92.55 €**
GranaSAT CanSat Total Electronic Cost	**83.94 €**
Difference	**8.61 €**

Table A.7 – *PLA material total cost*

The total cost of the GranaSAT CanSat is 8.61 €higher than the one the Czech engineers designed. That is due to the LCD we used (bigger and with touch sensor), and to the receiver radio and GPS antennas. Those antennas were not included on the Czech engineers CanSat. The rest of the modules are, in general, cheaper in our design.

6.3 Software

Software	Liscence Owner	Cost (€)
Altium	GranaSAT	Free (sponsorship)
SolidWorks®	GranaSAT	Free (sponsorship)
PCB Toolkit	Irene Gil	Free License
ArduinoIDE	Irene Gil	Free License
VSCode	Irene Gil	Free License

Continued on next page

Software	Liscence Owner	Cost (€)
MathCAD	Irene Gil	Free License
Python (Anaconda)	Irene Gil	Free License
ATMEL Studio	Irene Gil	Free License
Logic	Irene Gil	Free License
Pulse View	Irene Gil	Free License
OpenChoice (Textronix)	GranaSAT	Free (sponsorship)
VNA Software	GranaSAT	Free (sponsorship)
VS Paradigm	Irene Gil	Free License
TeXnicCenter	Irene Gil	Free License
Miktex	Irene Gil	Free License
SumatraPDF	Irene Gil	Free License

Table 6.8 – *Software cost*

6.4 Human Resources Cost

The development of this Bachelor's Thesis would have required hiring two people. The first one a **junior engineer** (10.00 €/h), hired as a full-time worker during six months. Secondly, as s Project Supervisor a **senior engineer** is hired (50.00 €/h), computing 5hours per week. Then, the Human Resources amounts to 7056.00 €, as detailed in Table 6.9.

Post	Time (Hours)	Cost (€)
Junior Engineer	1056.00	10560.00
Senior Engineer	120.00	6000.00
	TOTAL	**7056.00 €**

Table 6.9 – *Human Resources Cost*

6.5 Total Project Cost

Adding the cost of all sections (Section A.1,Section 6.3,Section 6.4) the total bachelor thesis budget is:

Section	Cost (€)
Hardware Budget	133.12
Software Budget	0.00

Continued on next page

Section	Cost (€)
Human Resources Budget	7056.00
Total Cost	**7189.12**

Table 6.10 – *Software cost*

Appendix B

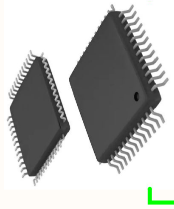
Altium Files

B.1 Czech CanSat Kit SCH

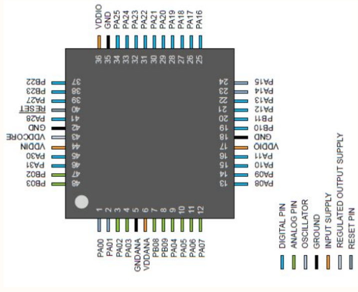
DATASHEET

FEATURES	SAMD21G18A-AUT
POWER SUPPLY	1.62 V to 3.63 V
CURRENT SUPPLY	46 mA
CRYSTAL OSCILLATOR	0.4 to 32 Mhz
OPERATING TEMPERATURE RANGE	- 40 ºC to 125 ºC
PROCESSOR	ARM CORTEX -M0+CPU running at up to 48 MHz: - Single-cycle hardware multiplier - Micro Trace Buffer (MTB)
MEMORY	- 256 in-system self-programmable Flash - 32 kB SRAM Memory
SYSTEM	- Power-on Reset (POR) and Brown-out Detection (BOD) -Internal and external clock option: 48 MHz Digital Frequency-Locked Loop (DFLL48M) and 48MHZ to 96 MHz Fractional Digital Phase-Locked Loop (FDPLL96M) - External Interrupt Controller (EIC) - 16 external interrupts - One Non-maskable Interrupt (NMI) - Two-pin Serial Wire Debug (SWD) programming , test and debugging interface
LOW POWER	- Idle and Stand-by Sleep modes - SleepWalking peripherals

FEATURES	SAMD21G18A-AUT
LOW POWER	- Idle and Stand-by Sleep modes - SleepWalking peripherals
PERIPHERALS	- 12- Channel Direct Memory Access Controller (DMAC) - 12-Channel Event System - Up to five 16-bit Timer/Counters (TC) - Up to four 24-bit Timer/Counters (TCC) - 32-bit Real Time Counter (RTC) with clock/calendar function - Watchdog Timer (WDT) - CRC-32 generator - One full-speed (12 Mbps) Universal Serial Bus (USB) 2.0 interface -Up to six Serial Communication Interfaces (SERCOM) => USART , I2C , SPI , LIN slave. - One two channel Inter-IC Sound (I2S) - One 12-bit , 350 kps Anlog-to-Digital Converter (ADC) with up to 20 channels - 10-bit , 350 kps Digital-to-Analog Converter (DAC) - Up to four Analog Comparators (AC) with Window Compare function - Peripheral Touch Controller (PTC)
DIMMENSION	TQFP package : 7 x 7 x 1.0 mm^3 QFN : 7 x 7 mm^2 body , 0.40 mm Terminals , 5.1 x 5.1 mm^2 EP
PRICE	2.73 € https://es.farnell.com/

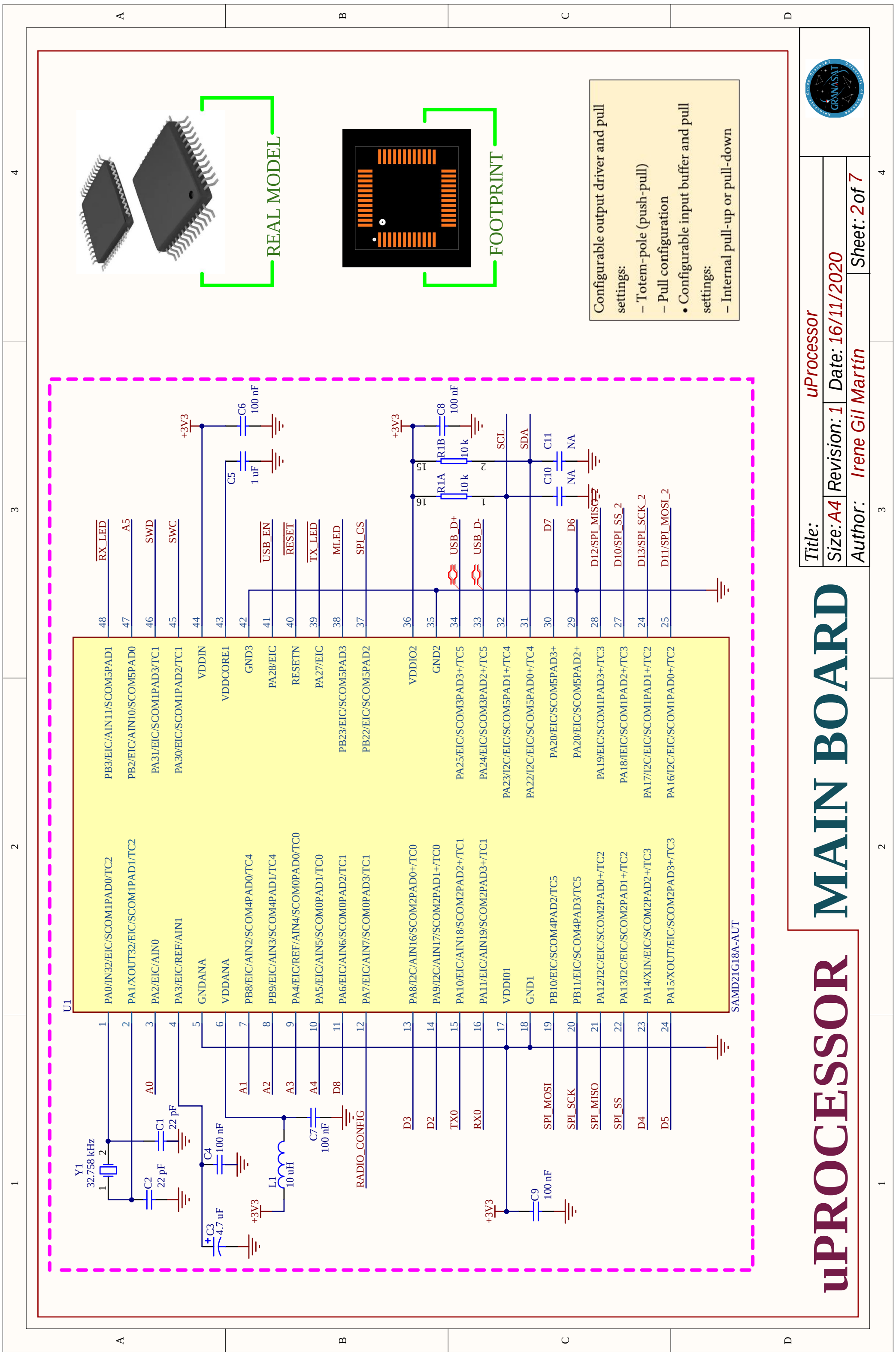


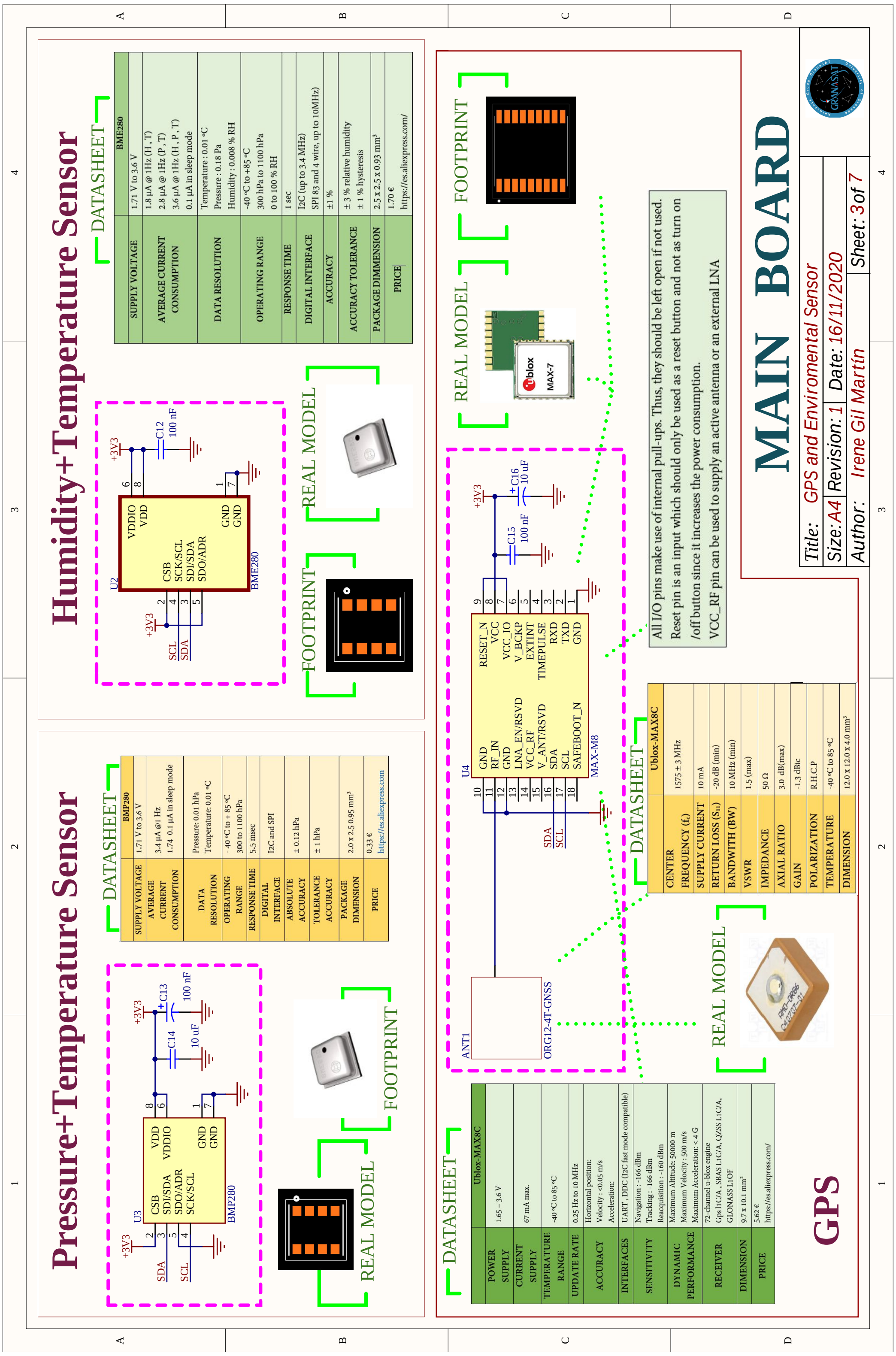
REAL MODEL

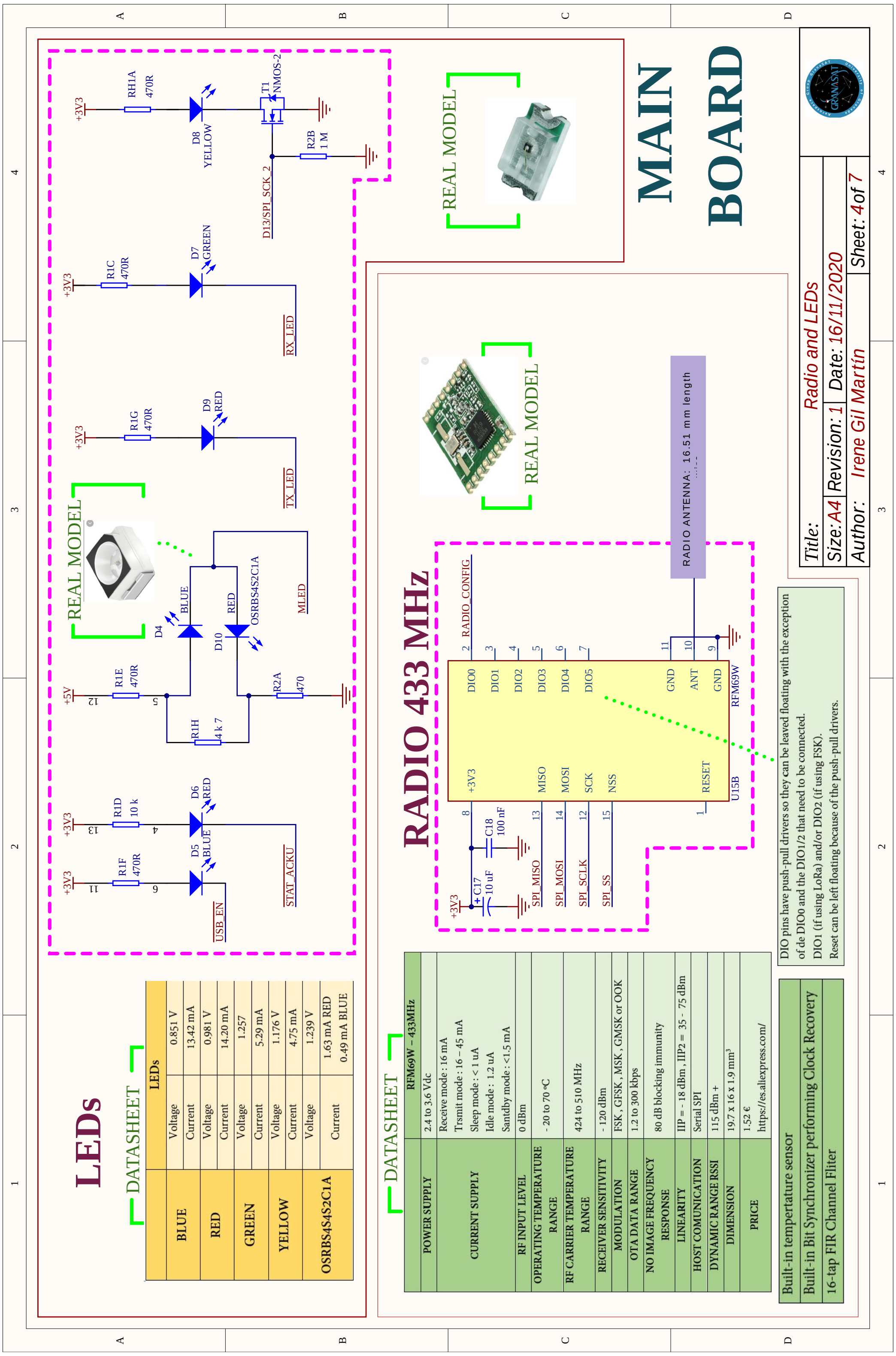


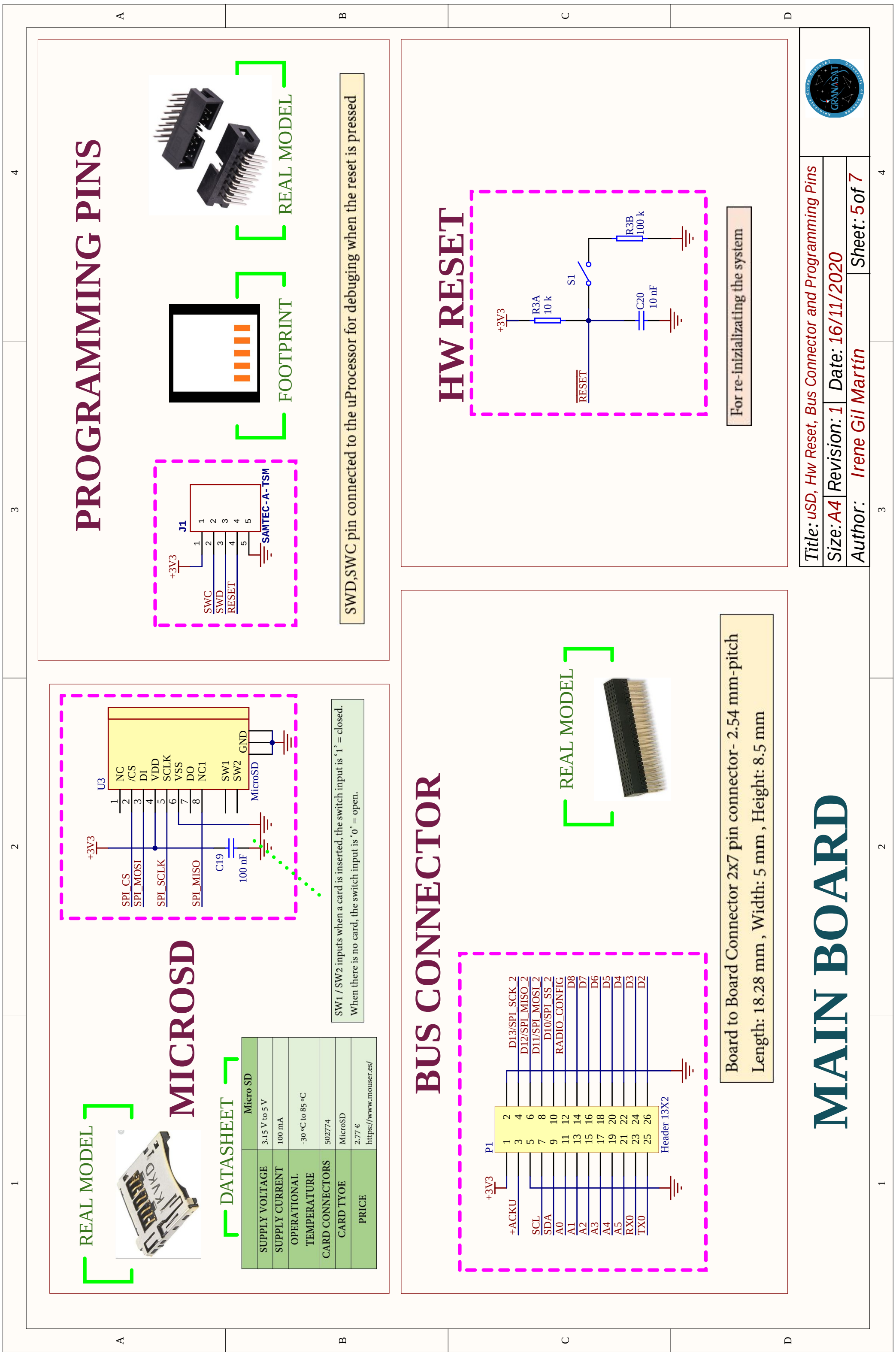
uPROCESSOR INFORMATION

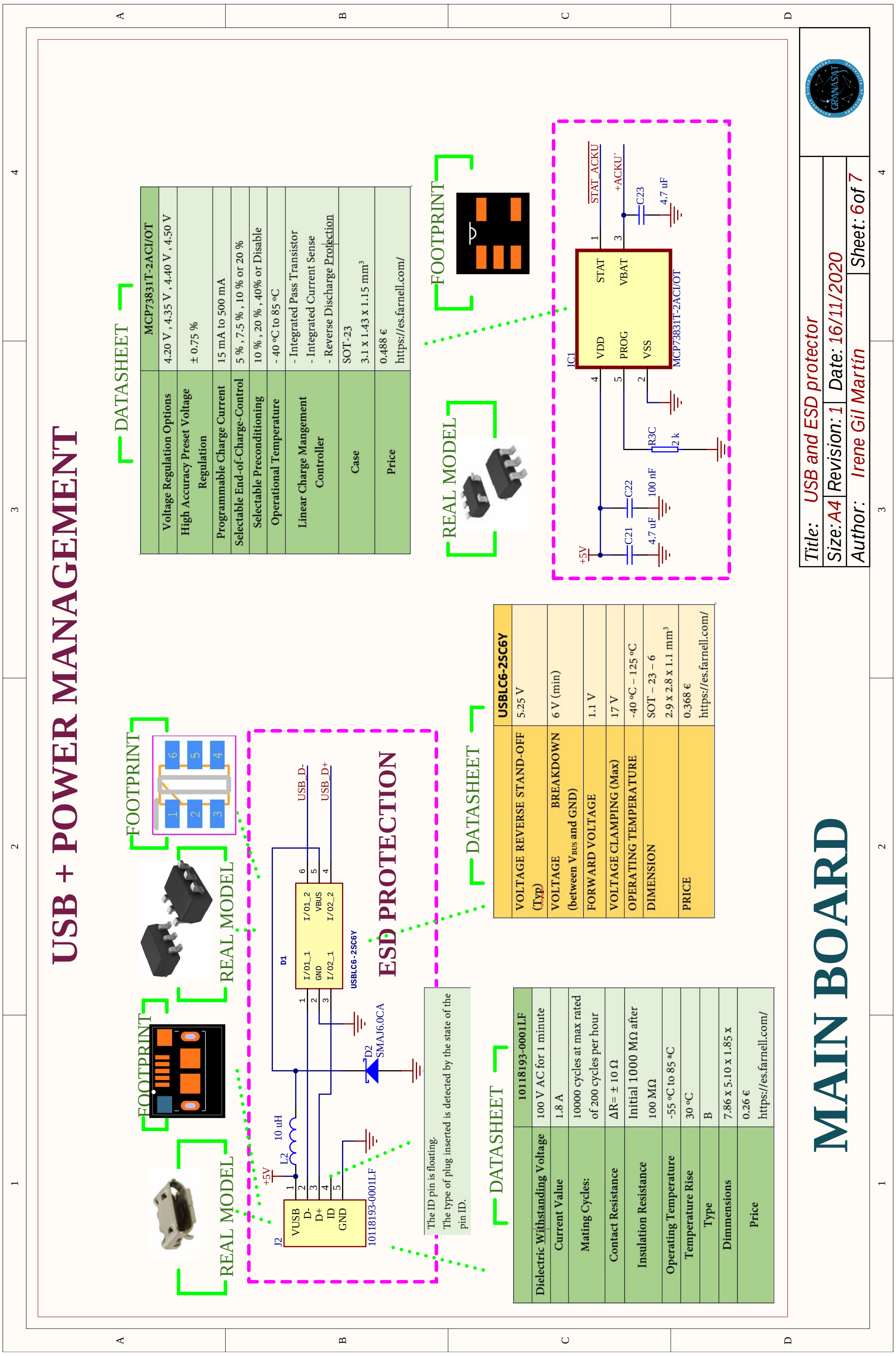
Title: uProcessor Information
Size: A4 Revision: 1 Date: 16/11/2020
Author: Irene Gil Martín Sheet: 1 of 7


uPROCESSOR MAIN BOARD
REAL MODEL
FOOTPRINT
Configurable output driver and pull settings:
– Totem-pole (push-pull)
– Pull configuration
• Configurable input buffer and pull settings:
– Internal pull-up or pull-down
Title: uProcessor
Size: A4 Revision: 1 Date: 16/11/2020
Author: Irene Gil Martin Sheet: 2 of 7

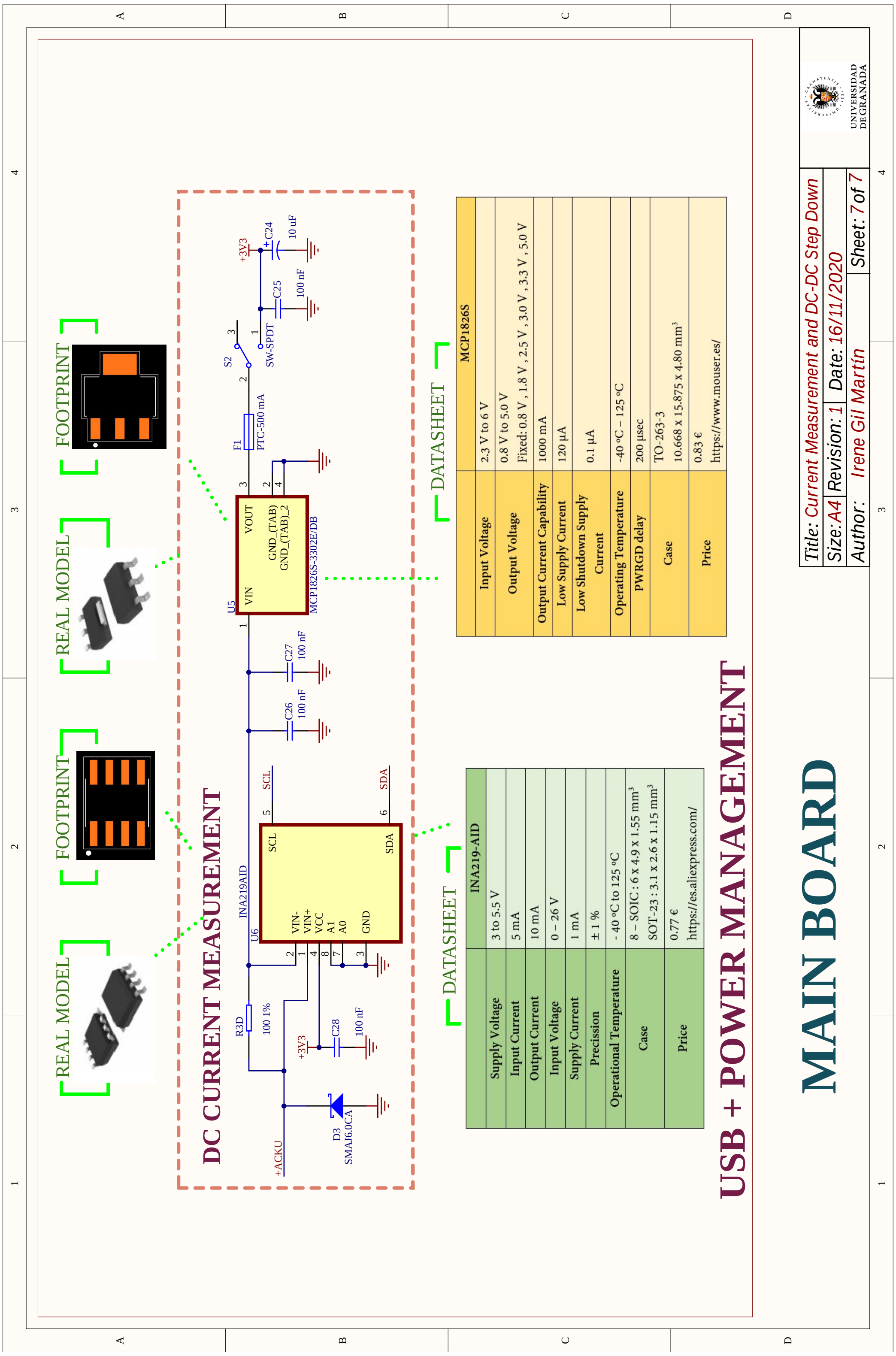

Pressure+Temperature Sensor
Humidity+Temperature Sensor
GPS
DATASHEET
FOOTPRINT
REAL MODEL
All I/O pins make use of internal pull-ups. Thus, they should be left open if not used.
Reset pin is an input which should only be used as a reset button and not as turn on /off button since it increases the power consumption.
VCC_RF pin can be used to supply an active antenna or an external LNA
MAIN BOARD
Title: GPS and Enviromental Sensor
Size: A4 Revision: 1 Date: 16/11/2020 Sheet: 3 of 7
Author: Irene Gil Martín

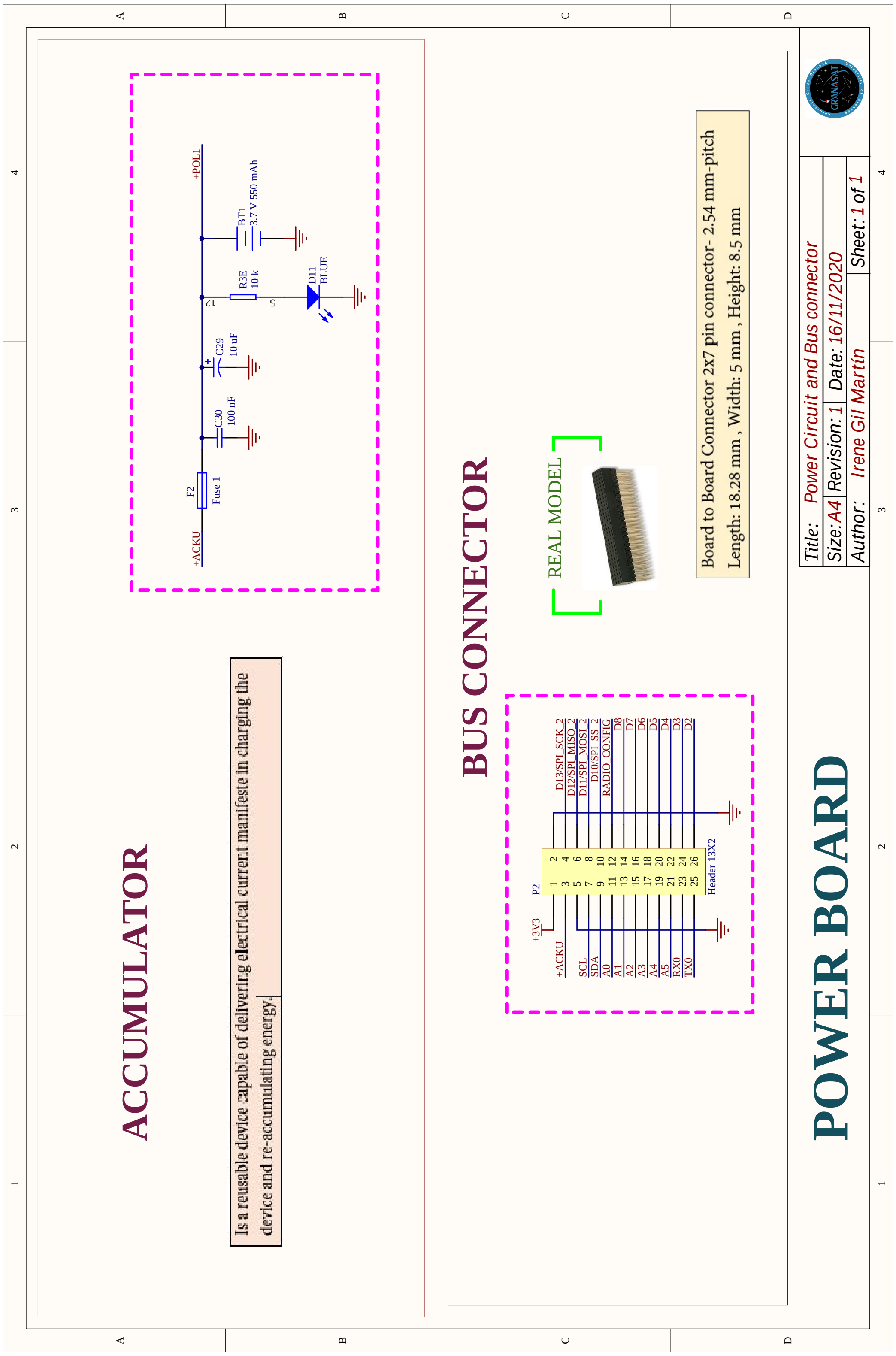

LEDs
DATASHEET
BLUE
RED
GREEN
YELLOW
OSRBS4S2C1A
LEDs
Voltage 0.851 V
Current 13.42 mA
Voltage 0.981 V
Current 14.20 mA
Voltage 1.257
Current 5.29 mA
Voltage 1.176 V
Current 4.75 mA
Voltage 1.239 V
Current 1.63 mA RED 0.49 mA BLUE
REAL MODEL
RADIO 433 MHz
DATASHEET
RFM69W - 433MHz
POWER SUPPLY 2.4 to 3.6 Vdc
CURRENT SUPPLY Receive mode : 16 mA Trsmit mode : 16 – 45 mA Sleep mode : < 1 uA Idle mode : 1.2 uA Santdby mode : <1.5 mA
RF INPUT LEVEL 0 dBm
OPERATING TEMPERATURE RANGE - 20 to 70 ºC
RF CARRIER TEMPERATURE RANGE 424 to 510 MHz
RECEIVER SENSITIVITY - 120 dBm
MODULATION FSK , GFSK , MSK , GMSK or OOK
OTA DATA RANGE 1.2 to 300 kbps
NO IMAGE FREQUENCY RESPONSE 80 dB blocking immunity
LINEARITY IIP = - 18 dBm , IIP2 = 35 - 75 dBm
HOST COMUNICATION Serial SPI
DYNAMIC RANGE RSSI 115 dBm +
DIMENSION 19.7 x 16 x 1.9 mm³
PRICE 1.52 € https://es.aliexpress.com/
Built-in tempertature sensor
Built-in Bit Synchronizer performing Clock Recovery
16-tap FIR Channel Fliter
DIO pins have push-pull drivers so they can be leaved floating with the exception of de DIO0 and the DIO1/2 that need to be connected.
DIO1 (if using LoRa) and/or DIO2 (if using FSK).
Reset can be left floating because of the push-pull drivers.
RADIO ANTENNA: 16.51 mm length
MAIN BOARD
Title: Radio and LEDs
Size: A4 Revision: 1 Date: 16/11/2020
Author: Irene Gil Martín Sheet: 4 of 7

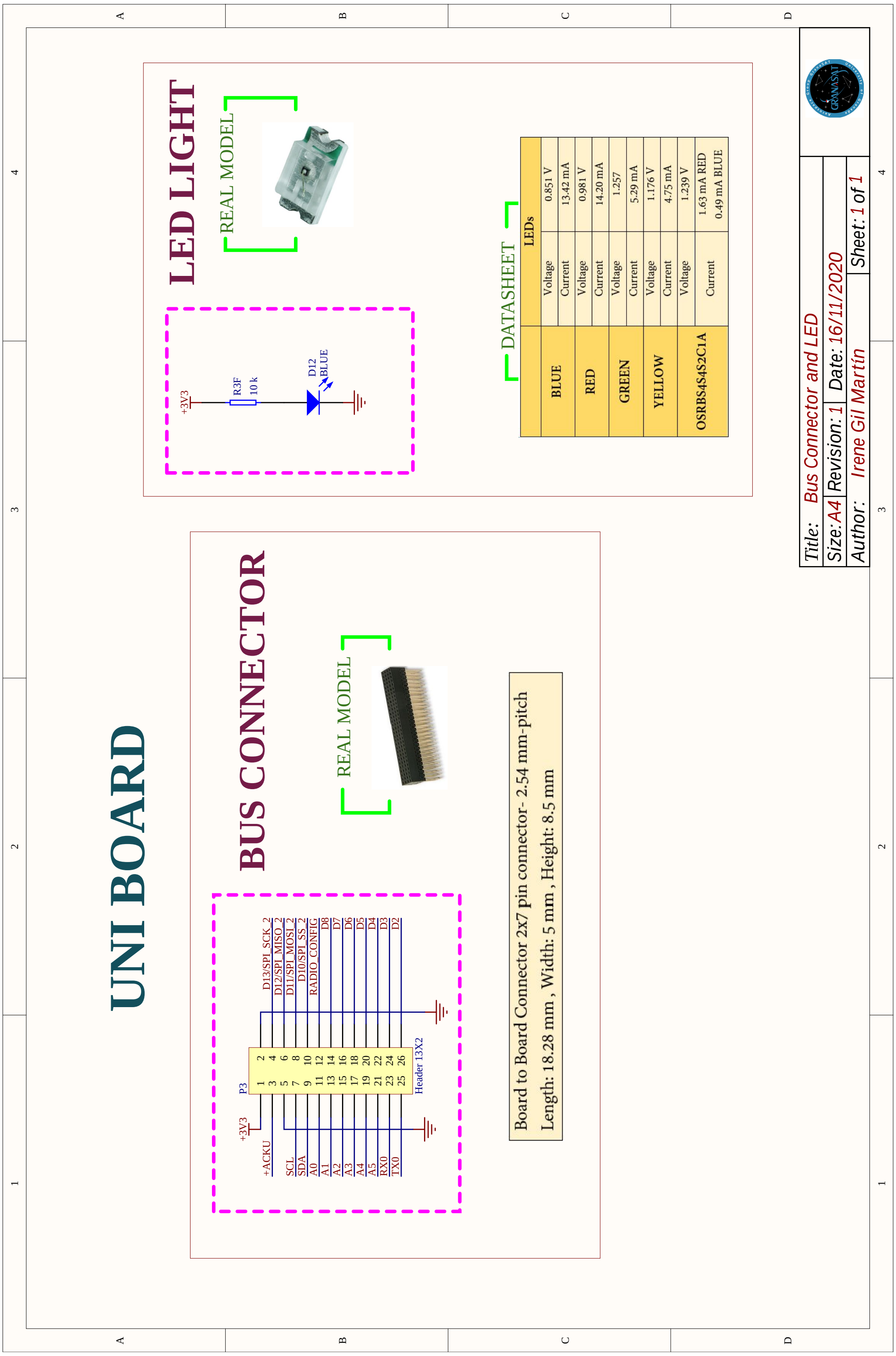

PROGRAMMING PINS
REAL MODEL
FOOTPRINT
+3V3
J1
SWC
SWD
RESET
SAMTEC-A-TSM
SWD,SWC pin connected to the uProcessor for debuging when the reset is pressed
HW RESET
+3V3
R3A
10 k
S1
R3B
100 k
C20
10 nF
RESET
For re-inizializating the system
Title: uSD, Hw Reset, Bus Connector and Programming Pins
Size: A4
Revision: 1
Date: 16/11/2020
Author: Irene Gil Martín
Sheet: 5 of 7
MICROSD
REAL MODEL
DATASHEET
+3V3
U3
NC
/CS
DI
VDD
SCLK
VSS
DO
NC1
SW1
SW2
GND
MicroSD
SPI_CS
SPI_MOSI
SPI_SCLK
SPI_MISO
C19
100 nF
SW1 / SW2 inputs when a card is inserted, the switch input is '1' = closed.
When there is no card, the switch input is '0' = open.
Micro SD
SUPPLY VOLTAGE
3.15 V to 5 V
SUPPLY CURRENT
100 mA
OPERATIONAL TEMPERATURE
-30 ºC to 85 ºC
CARD CONNECTORS
502774
CARD TYOE
MicroSD
PRICE
2.77 €
https://www.mouser.es/
BUS CONNECTOR
REAL MODEL
+3V3
P1
+ACKU
SCL
SDA
A0
A1
A2
A3
A4
A5
RX0
TX0
D13/SPI_SCK_2
D12/SPI_MISO_2
D11/SPI_MOSI_2
D10/SPI_SS_2
RADIO_CONFIG
D8
D7
D6
D5
D4
D3
D2
Header 13X2
Board to Board Connector 2x7 pin connector- 2.54 mm-pitch
Length: 18.28 mm , Width: 5 mm , Height: 8.5 mm
MAIN BOARD

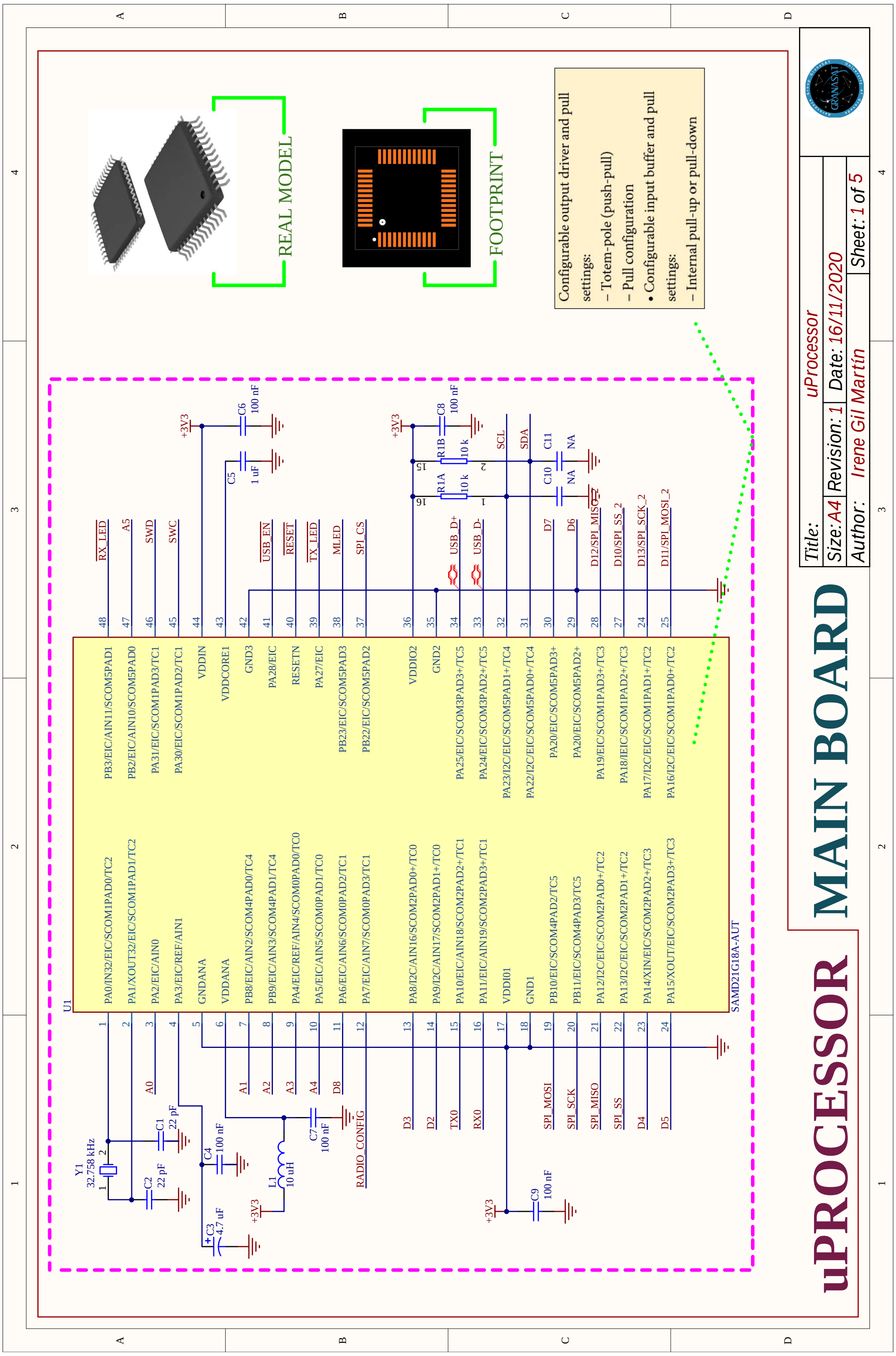


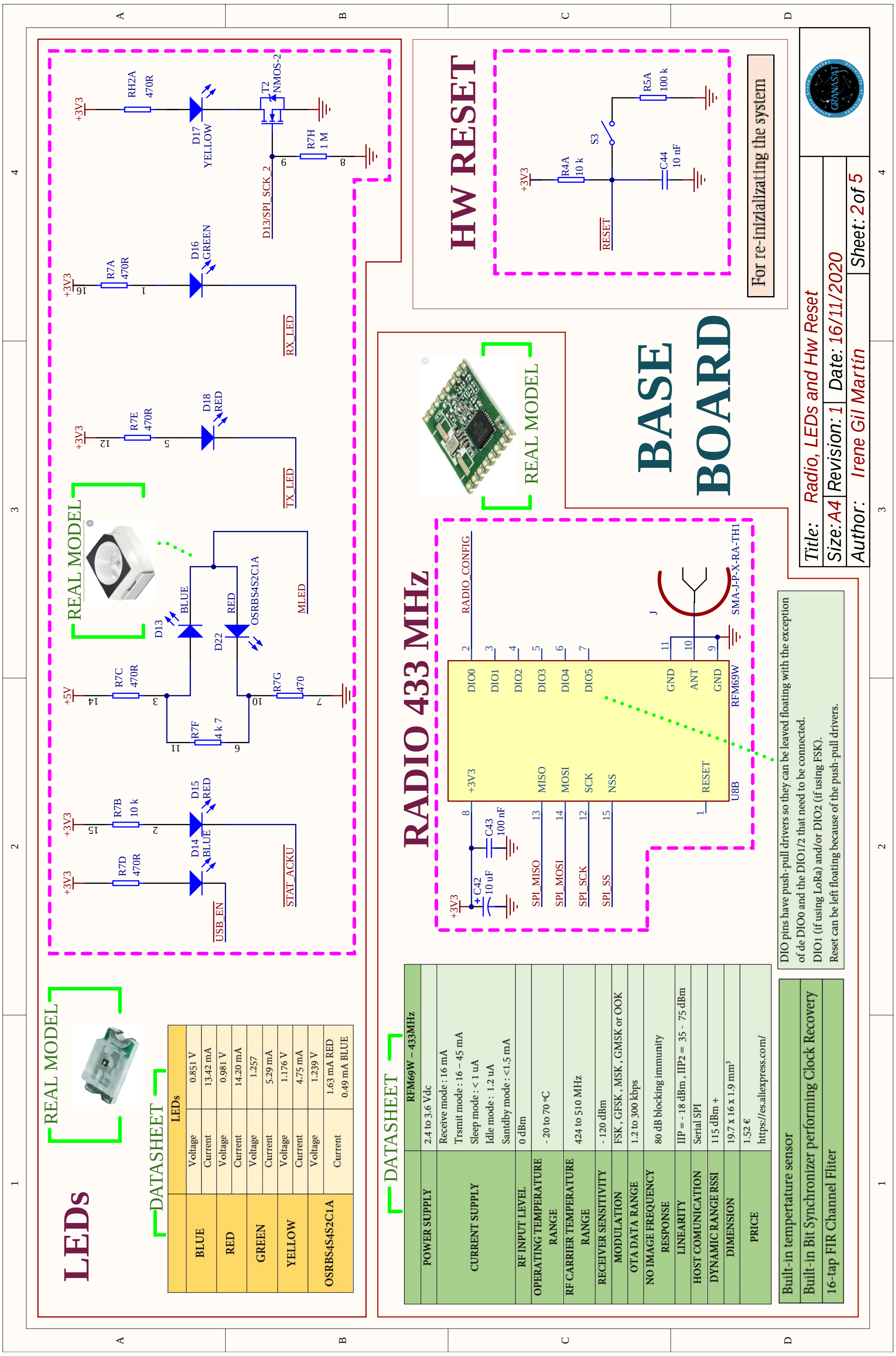
USB + POWER MANAGEMENT
ESD PROTECTION
DATASHEET
FOOTPRINT
REAL MODEL
MAIN BOARD
Title: USB and ESD protector
Size: A4
Revision: 1
Date: 16/11/2020
Author: Irene Gil Martin
Sheet: 6 of 7
MCP73831T-2ACI/OT
USBLC6-2SC6Y
10118193-0001LF
SMAJ6.0CA
The ID pin is floating.
The type of plug inserted is detected by the state of the pin ID.


REAL MODEL
FOOTPRINT
REAL MODEL
FOOTPRINT
DC CURRENT MEASUREMENT
+ACKU
D3
SMAJ6.0CA
R3D
100 1%
+3V3
C28
100 nF
U6
INA219AID
VIN-
VIN+
VCC
A1
A0
GND
SCL
SDA
C26
100 nF
C27
100 nF
U5
VIN
VOUT
GND_(TAB)
GND_(TAB)_2
MCP1826S-3302E/DB
F1
PTC-500 mA
S2
SW-SPDT
C25
100 nF
C24
10 uF
DATASHEET
INA219-AID
Supply Voltage 3 to 5.5 V
Input Current 5 mA
Output Current 10 mA
Input Voltage 0 – 26 V
Supply Current 1 mA
Precission ± 1 %
Operational Temperature - 40 ºC to 125 ºC
Case 8 – SOIC : 6 x 4.9 x 1.55 mm³ SOT-23 : 3.1 x 2.6 x 1.15 mm³
Price 0.77 € https://es.aliexpress.com/
DATASHEET
MCP1826S
Input Voltage 2.3 V to 6 V
Output Voltage 0.8 V to 5.0 V Fixed: 0.8 V , 1.8 V , 2.5 V , 3.0 V , 3.3 V , 5.0 V
Output Current Capability 1000 mA
Low Supply Current 120 µA
Low Shutdown Supply Current 0.1 µA
Operating Temperature -40 ºC – 125 ºC
PWRGD delay 200 µsec
Case TO-263-3 10.668 x 15.875 x 4.80 mm³
Price 0.83 € https://www.mouser.es/
USB + POWER MANAGEMENT
MAIN BOARD
Title: Current Measurement and DC-DC Step Down
Size: A4 Revision: 1 Date: 16/11/2020
Author: Irene Gil Martín Sheet: 7 of 7
UNIVERSIDAD DE GRANADA

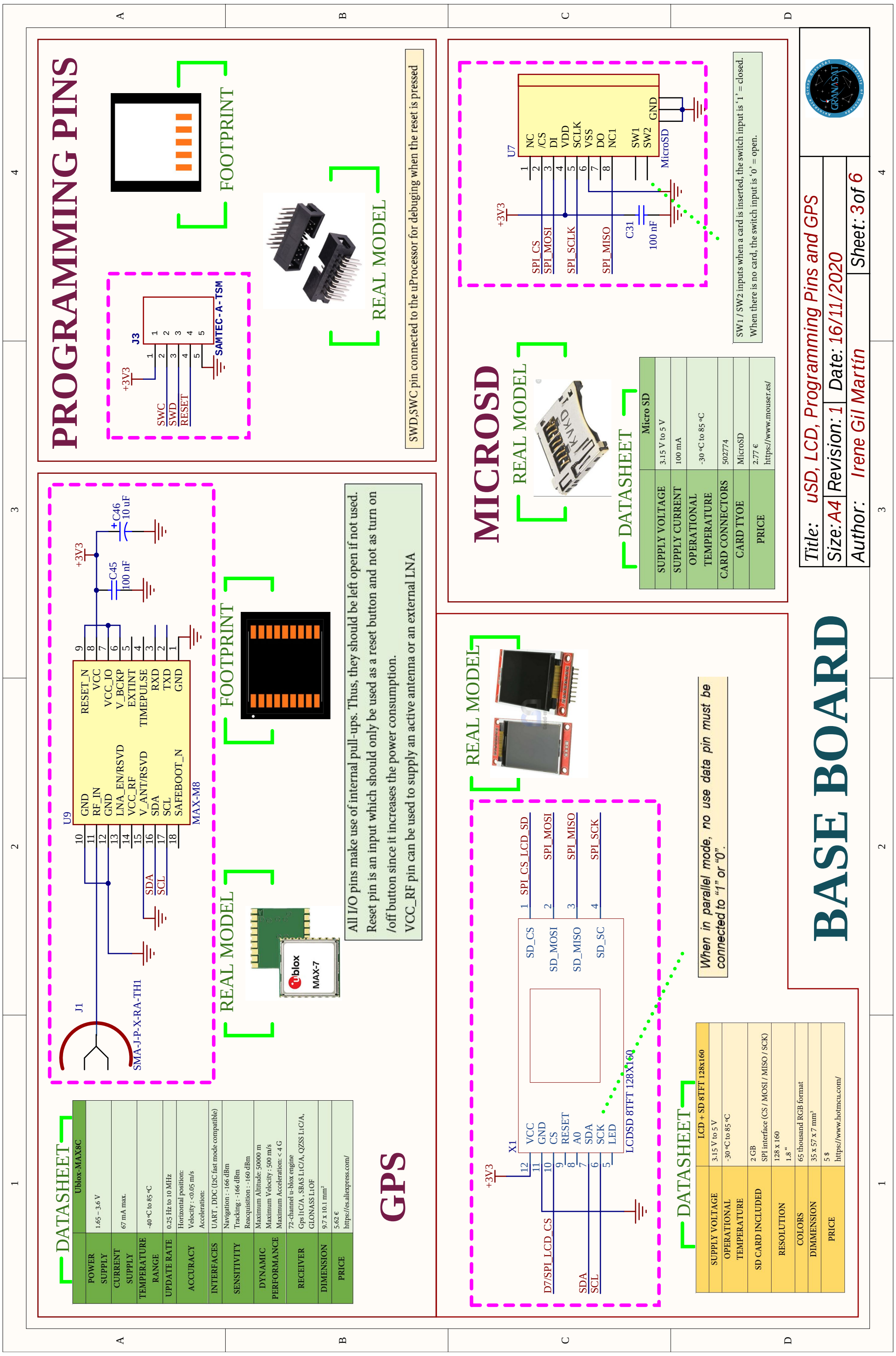

ACCUMULATOR
Is a reusable device capable of delivering electrical current manifeste in charging the device and re-accumulating energy.
+ACKU
F2
Fuse 1
C30
100 nF
C29
10 uF
R3E
10 k
BT1
3.7 V 550 mAh
+POL1
D11
BLUE
BUS CONNECTOR
REAL MODEL
Board to Board Connector 2x7 pin connector - 2.54 mm-pitch
Length: 18.28 mm , Width: 5 mm , Height: 8.5 mm
+3V3
+ACKU
P2
Header 13X2
SCL
SDA
A0
A1
A2
A3
A4
A5
RX0
TX0
D13/SPI_SCK_2
D12/SPI_MISO_2
D11/SPI_MOSI_2
D10/SPI_SS_2
RADIO_CONFIG
D8
D7
D6
D5
D4
D3
D2
POWER BOARD
Title: Power Circuit and Bus connector
Size: A4
Revision: 1
Date: 16/11/2020
Author: Irene Gil Martín
Sheet: 1 of 1

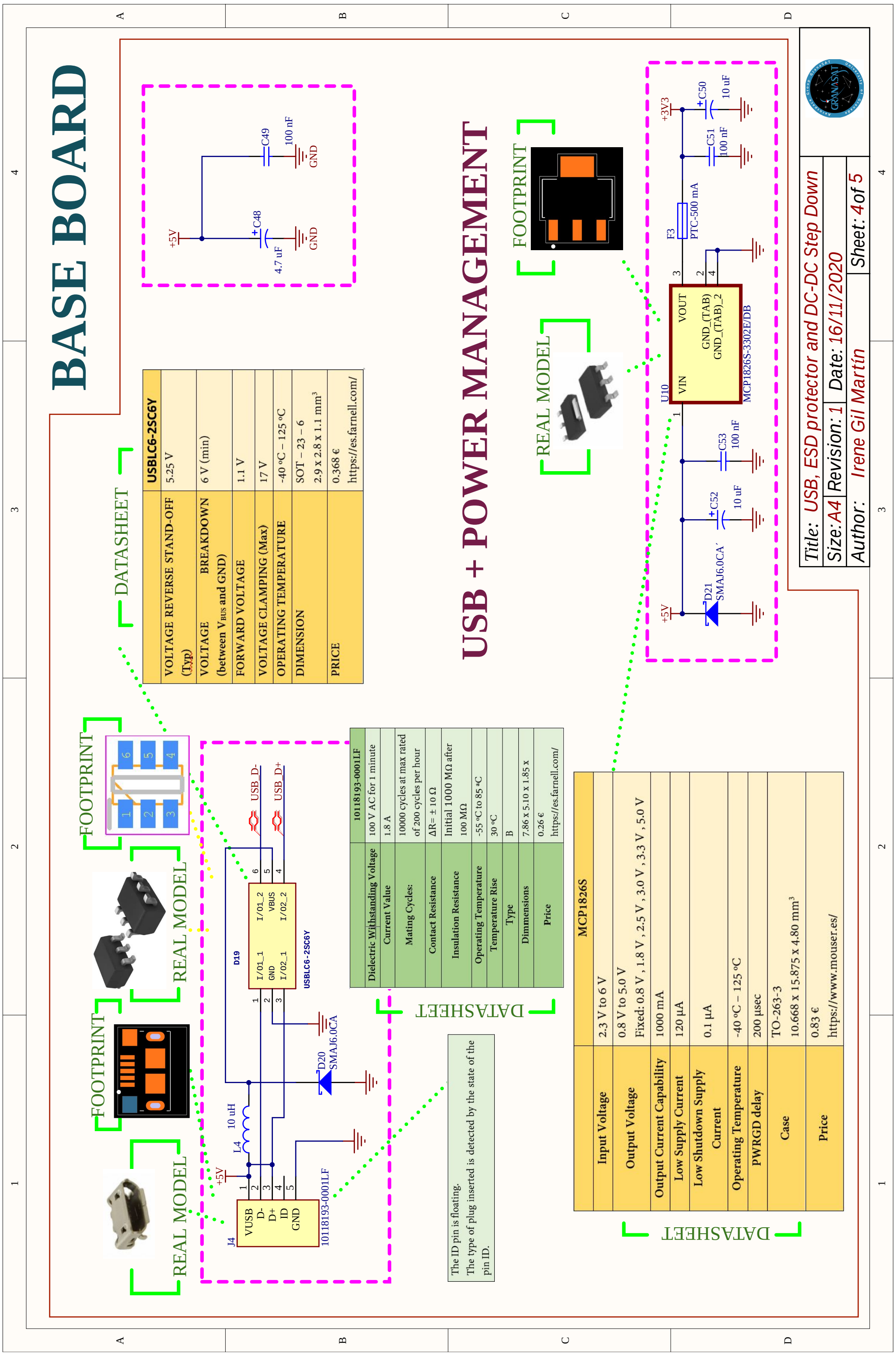

UNI BOARD
BUS CONNECTOR
+3V3
P3
+ACKU
SCL
SDA
A0
A1
A2
A3
A4
A5
RX0
TX0
D13/SPI_SCK_2
D12/SPI_MISO_2
D11/SPI_MOSI_2
D10/SPI_SS_2
RADIO_CONFIG
D8
D7
D6
D5
D4
D3
D2
Header 13X2
REAL MODEL
Board to Board Connector 2x7 pin connector- 2.54 mm-pitch
Length: 18.28 mm , Width: 5 mm , Height: 8.5 mm
LED LIGHT
+3V3
R3F
10 k
D12
BLUE
REAL MODEL
DATASHEET
LEDs
BLUE
Voltage 0.851 V
Current 13.42 mA
RED
Voltage 0.981 V
Current 14.20 mA
GREEN
Voltage 1.257
Current 5.29 mA
YELLOW
Voltage 1.176 V
Current 4.75 mA
OSRBS4S4S2C1A
Voltage 1.239 V
Current 1.63 mA RED 0.49 mA BLUE
Title: Bus Connector and LED
Size: A4 Revision: 1 Date: 16/11/2020
Author: Irene Gil Martín Sheet: 1 of 1

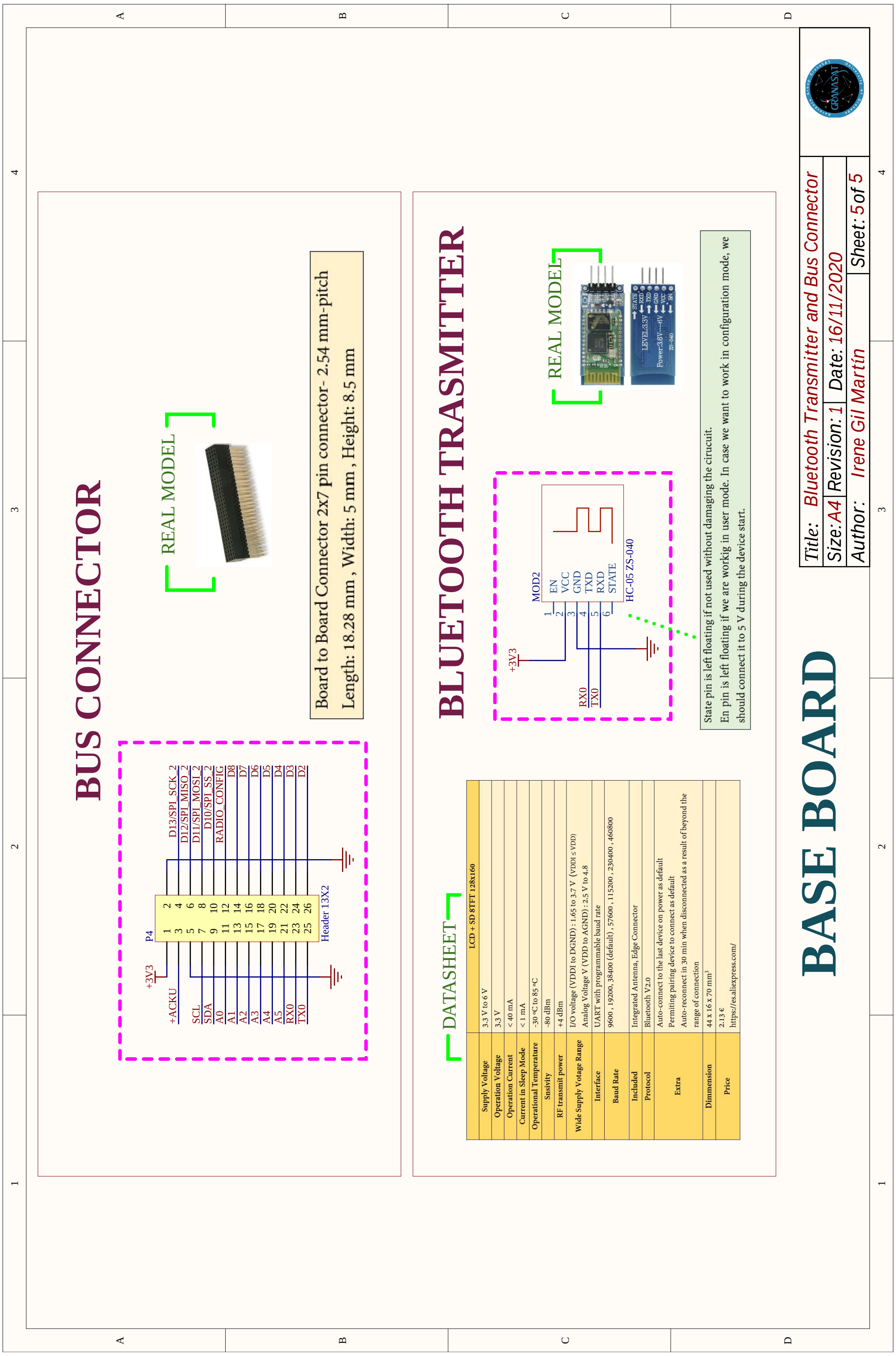

uPROCESSOR MAIN BOARD
REAL MODEL
FOOTPRINT
Configurable output driver and pull settings:
– Totem-pole (push-pull)
– Pull configuration
• Configurable input buffer and pull settings:
– Internal pull-up or pull-down
SAMD21G18A-AUT
Title: uProcessor
Size: A4
Revision: 1
Date: 16/11/2020
Author: Irene Gil Martin
Sheet: 1 of 5



LEDs
REAL MODEL
DATASHEET
RADIO 433 MHz
HW RESET
For re-inizializating the system
BASE BOARD
Title: Radio, LEDs and Hw Reset
Size: A4 Revision: 1 Date: 16/11/2020
Author: Irene Gil Martín
Sheet: 2 of 5
Built-in tempertature sensor
Built-in Bit Synchronizer performing Clock Recovery
16-tap FIR Channel Fliter
DIO pins have push-pull drivers so they can be leaved floating with the exception of de DIO0 and the DIO1/2 that need to be connected.
DIO1 (if using LoRa) and/or DIO2 (if using FSK).
Reset can be left floating because of the push-pull drivers.

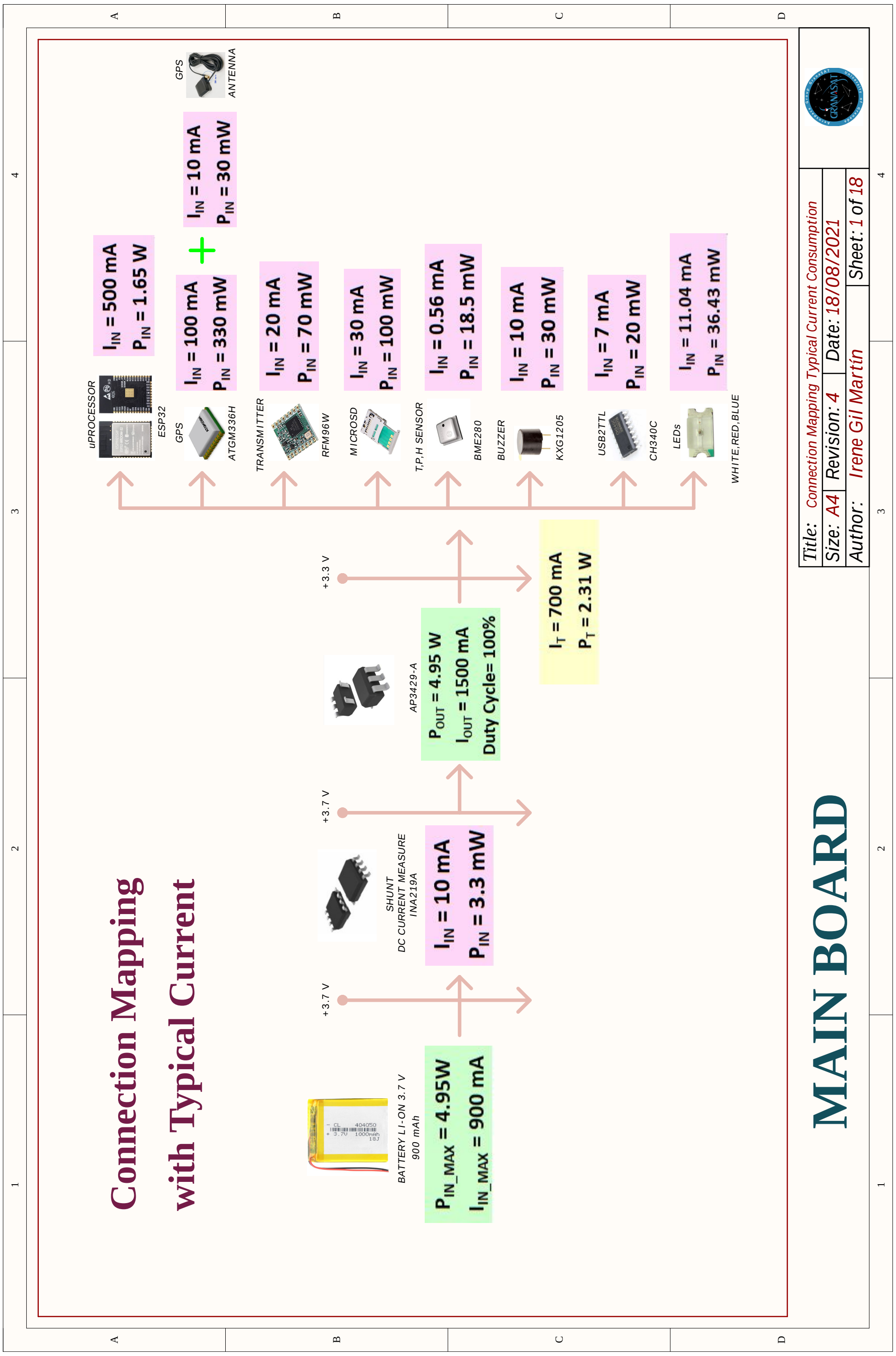

PROGRAMMING PINS
FOOTPRINT
REAL MODEL
J3
SAMTEC-A-TSM
SWC
SWD
RESET
SWD,SWC pin connected to the uProcessor for debuging when the reset is pressed
MICROSD
U7
NC
/CS
DI
VDD
SCLK
VSS
DO
NC1
SW1
SW2
GND
MicroSD
SPI_CS
SPI_MOSI
SPI_SCLK
SPI_MISO
C31
100 nF
SW1 / SW2 inputs when a card is inserted, the switch input is '1' = closed.
When there is no card, the switch input is '0' = open.
DATASHEET
Micro SD
SUPPLY VOLTAGE
3.15 V to 5 V
SUPPLY CURRENT
100 mA
OPERATIONAL TEMPERATURE
-30 ºC to 85 ºC
CARD CONNECTORS
502774
CARD TYOE
MicroSD
PRICE
2.77 €
https://www.mouser.es/
GPS
U9
MAX-M8
J1
SMA-J-P-X-RA-TH1
C45
100 nF
C46
10 uF
+3V3
All I/O pins make use of internal pull-ups. Thus, they should be left open if not used.
Reset pin is an input which should only be used as a reset button and not as turn on /off button since it increases the power consumption.
VCC_RF pin can be used to supply an active antenna or an external LNA
DATASHEET
Ublox-MAX8C
POWER SUPPLY
1.65 – 3.6 V
CURRENT SUPPLY
67 mA max.
TEMPERATURE RANGE
-40 ºC to 85 ºC
UPDATE RATE
0.25 Hz to 10 MHz
ACCURACY
INTERFACES
SENSITIVITY
DYNAMIC PERFORMANCE
RECEIVER
DIMENSION
9.7 x 10.1 mm
PRICE
5.62 €
BASE BOARD
X1
LCDSD 8TFT 128X160
SD_CS
SD_MOSI
SD_MISO
SD_SC
SPI_CS_LCD_SD
SPI_MOSI
SPI_MISO
SPI_SCK
D7/SPI_LCD_CS
SDA
SCL
When in parallel mode, no use data pin must be connected to "1" or "0".
DATASHEET
LCD + SD 8TFT 128x160
SUPPLY VOLTAGE
3.15 V to 5 V
OPERATIONAL TEMPERATURE
-30 ºC to 85 ºC
SD CARD INCLUDED
2 GB
RESOLUTION
128 x 160
COLORS
65 thousand RGB format
DIMENSION
35 x 57 x 7 mm
PRICE
5 $
Title: uSD, LCD, Programming Pins and GPS
Size: A4
Revision: 1
Date: 16/11/2020
Author: Irene Gil Martin
Sheet: 3 of 6

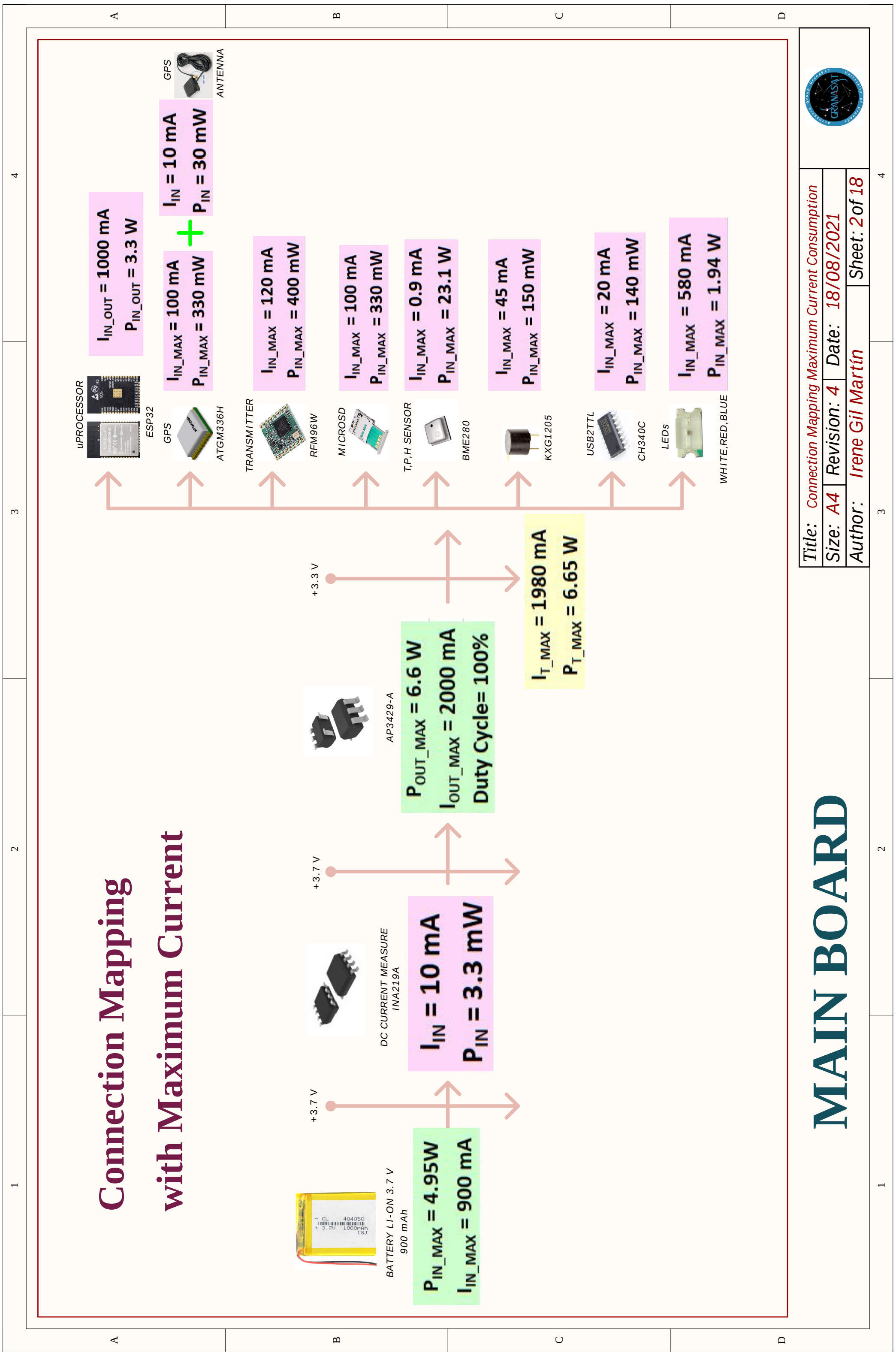

BASE BOARD
USB + POWER MANAGEMENT
FOOTPRINT
REAL MODEL
DATASHEET
USBLC6-2SC6Y
MCP1826S
10118193-0001LF
The ID pin is floating.
The type of plug inserted is detected by the state of the pin ID.
Title: USB, ESD protector and DC-DC Step Down
Size: A4 Revision: 1 Date: 16/11/2020
Author: Irene Gil Martín Sheet: 4 of 5


BUS CONNECTOR
REAL MODEL
Board to Board Connector 2x7 pin connector- 2.54 mm-pitch
Length: 18.28 mm , Width: 5 mm , Height: 8.5 mm
P4
Header 13X2
+3V3
+ACKU
SCL
SDA
D13/SPI_SCK_2
D12/SPI_MISO_2
D11/SPI_MOSI_2
D10/SPI_SS_2
RADIO_CONFIG
BLUETOOTH TRASMITTER
REAL MODEL
MOD2
EN
VCC
GND
TXD
RXD
STATE
HC-05 ZS-040
RX0
TX0
State pin is left floating if not used without damaging the circuit.
En pin is left floating if we are working in user mode. In case we want to work in configuration mode, we should connect it to 5 V during the device start.
DATASHEET
BASE BOARD
Title: Bluetooth Transmitter and Bus Connector
Size: A4
Revision: 1
Date: 16/11/2020
Author: Irene Gil Martín
Sheet: 5 of 5

B.2 GranaSAT CanSat Kit SCH

B.2.1 Main Board SCH




Connection Mapping
with Maximum Current
MAIN BOARD
BATTERY LI-ON 3.7 V
900 mAh
P_IN_MAX = 4.95W
I_IN_MAX = 900 mA
+3.7 V
DC CURRENT MEASURE
INA219A
I_IN = 10 mA
P_IN = 3.3 mW
+3.7 V
AP3429-A
P_OUT_MAX = 6.6 W
I_OUT_MAX = 2000 mA
Duty Cycle= 100%
+3.3 V
I_T_MAX = 1980 mA
P_T_MAX = 6.65 W
uPROCESSOR
ESP32
I_IN_OUT = 1000 mA
P_IN_OUT = 3.3 W
GPS
ATGM336H
I_IN_MAX = 100 mA
P_IN_MAX = 330 mW
GPS
ANTENNA
I_IN = 10 mA
P_IN = 30 mW
TRANSMITTER
RFM96W
I_IN_MAX = 120 mA
P_IN_MAX = 400 mW
MICROSD
I_IN_MAX = 100 mA
P_IN_MAX = 330 mW
T,P,H SENSOR
BME280
I_IN_MAX = 0.9 mA
P_IN_MAX = 23.1 W
KXG1205
I_IN_MAX = 45 mA
P_IN_MAX = 150 mW
USB2TTL
CH340C
I_IN_MAX = 20 mA
P_IN_MAX = 140 mW
LEDs
WHITE,RED,BLUE
I_IN_MAX = 580 mA
P_IN_MAX = 1.94 W
Title: Connection Mapping Maximum Current Consumption
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 2 of 18

Connection Mapping

uProcessor Pin	Conexion to Module	Information
LED_TX_RX	Blue and White LED White -> ‘0’ Blue -> ‘1’	A blue led is illuminated when the uProcessor is receiving information through the RX0 pin. A white led is illuminated when the uProcessor is sending information through the TX0 pin.
LED_RADIO	White LED -> ‘0’	A white led is illuminated when the radio is transmitting
LED_3V3	Blue LED -> ‘0’	A blue led is illuminated when the uProcessor is connected to the power board, and it is receiving 3.3 V (only if it is charged)
LED_USB_CONNECTED	Red LED -> ‘0’	A red led is illuminated if the USB is connected.
LED_1PPS	Blue LED -> ‘0’	A blue led is illuminated every second marked by the 1ppps pin of the GPS module
LED_UNIBOARD	Blue and White LED White -> ‘0’ Blue -> ‘1’	A blue and white is illuminated in the UniBoard.
SPI_SCLK	MicroSD	Connection through SPI serial clock between the uProcessor and the microSD/Radio.
	Radio	
SPI_MISO	MicroSD	Connection through SPI master in slave out between the uProcessor and the microSD/Radio.
	Radio	
SPI_MOSI	MicroSD	Connection through SPI master out slave in between the uProcessor and the microSD/Radio.
	Radio	
SPI_SS	MicroSD	Connection through SPI slave selection between the uProcessor and the microSD/Radio.
	Radio	
RESET_RADIO	Radio -> ‘0’	Resetting pin for the Radio Module.
RESET_GPS	GPS -> ‘0’	Resetting pin for the GPS Module.
SHUTOFF_CONTROL_GPS	GPS	Shutoff control of module, low level effectively

uProcessor Pin	Conexion to Module	Information
SDA	T+P+H sensor	Connection through I2C bus to send information between the GPS and the uProcessor and between the P+T+H sensor and the uProcessor
SCL	T+P+H sensor	Connection through I2C bus to synchronize the GPS and the uProcessor to send information through the SDA and between the P+T+H sensor and the uProcessor to send information through the SDA
SW_TXD	GPS	Serial interface bus to transmit data (UART) from the uProcessor to the GPS or vice versa, from the uProcessor to de CH340C or vice versa
TXD0	CH340C	
SW_RXD	GPS	Serial interface bus to receive data (UART) from the uProcessor to the GPS or vice versa, from the uProcessor to de CH340C or vice versa
RXD0	CH340C	
SW_CONFIG_RADIO	RADIO	Pin connected for software configuration
1PPS	GPS	A 1 pulse per second pulse emitted by the GPS
GPIO0	Programming	Pin used for programming the module
BUZZER	Electromagnetic Buzzer	A buzzer signal activated for emitting a sound
GPIO18	UniBoard	Floating pin for future connections on the UniBoard
CARD_DETECT	MicroSD	Pin in order to detect when there is a card inserted.
+3V3	All	A 3V3 tension given by the buck converter which is responsible of converting the 3.7 V given by the battery to 3V3 for supplying all the modules
GND	All	Ground connection

USB PIN	Module	Information
D_P	CH340C	Differential signal from the USB positive port
D_N	CH340C	Differential signal from the USB negative port
+5V	Power Board	+5V tension given from the USB to feed the charger circuit
GND	All	Ground Connection

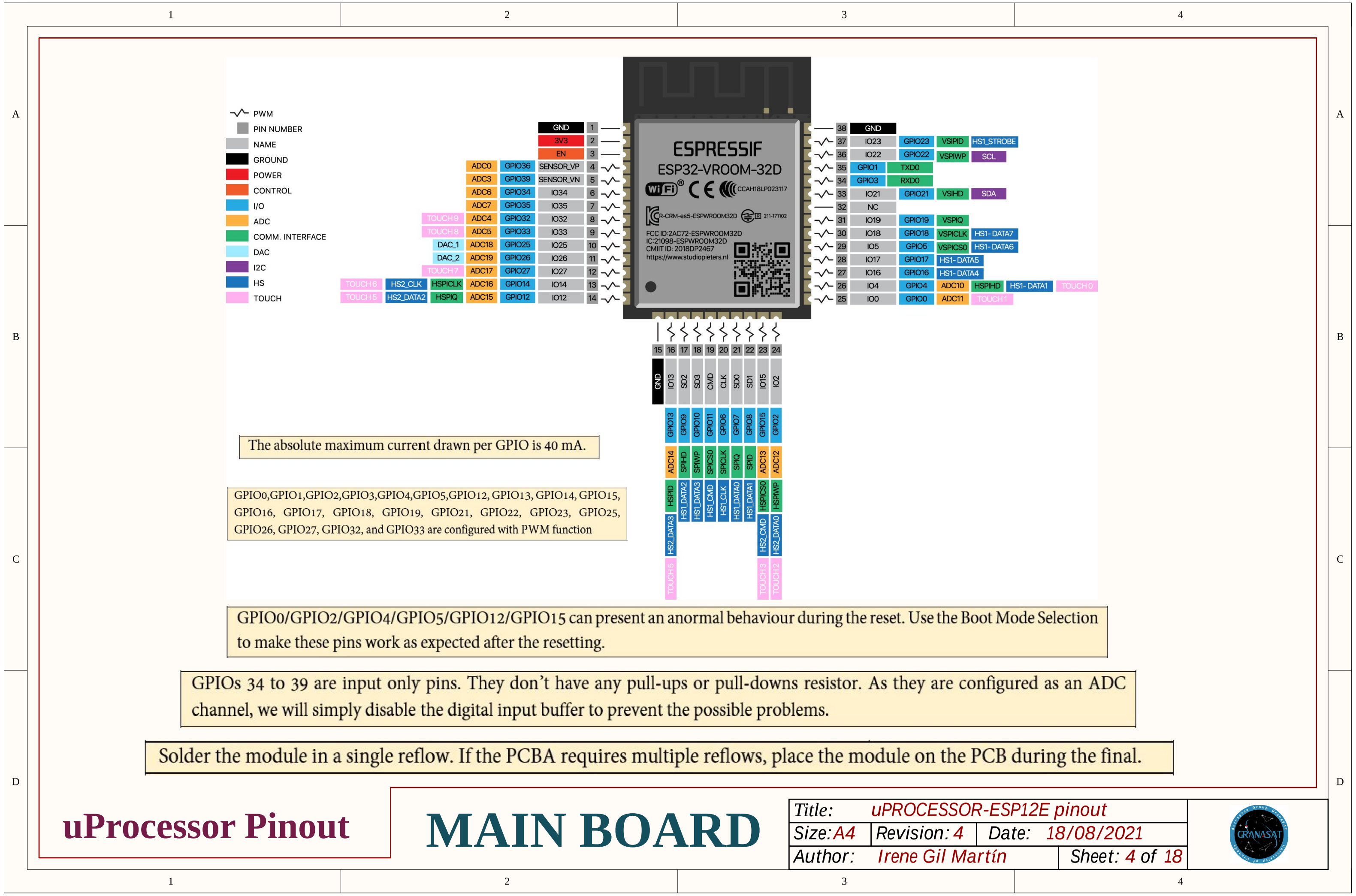
Battery PIN	Module	Information
BAT	INA 219A	3.7 V tension given by the battery

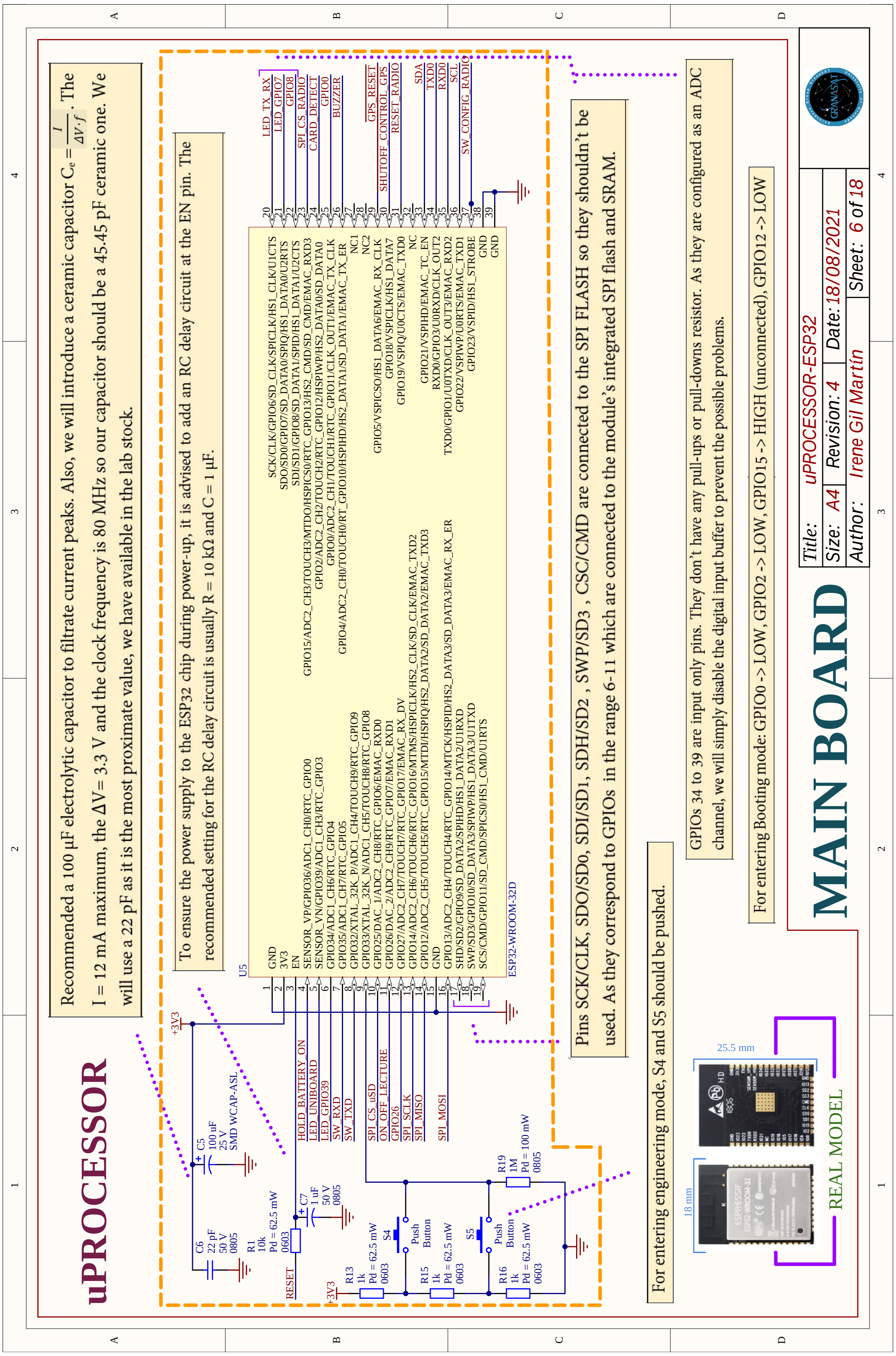
Battery PIN	Module	Information
BAT	INA 219A	3.7 V tension given by the battery

uProcessor Pin	Information
+3V3	A 3V3 tension given by the buck converter which is responsible of converting the 3.7 V given by the battery to 3V3 for supplying all the modules

MAIN BOARD

Title: Connection Mapping
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martín Sheet: 3 of 18


PWM
PIN NUMBER
NAME
GROUND
POWER
CONTROL
I/O
ADC
COMM. INTERFACE
DAC
I2C
HS
TOUCH
ESPRESSIF
ESP32-VROOM-32D
The absolute maximum current drawn per GPIO is 40 mA.
GPIO0,GPIO1,GPIO2,GPIO3,GPIO4,GPIO5,GPIO12, GPIO13, GPIO14, GPIO15, GPIO16, GPIO17, GPIO18, GPIO19, GPIO21, GPIO22, GPIO23, GPIO25, GPIO26, GPIO27, GPIO32, and GPIO33 are configured with PWM function
GPIO0/GPIO2/GPIO4/GPIO5/GPIO12/GPIO15 can present an anormal behaviour during the reset. Use the Boot Mode Selection to make these pins work as expected after the resetting.
GPIOs 34 to 39 are input only pins. They don't have any pull-ups or pull-downs resistor. As they are configured as an ADC channel, we will simply disable the digital input buffer to prevent the possible problems.
Solder the module in a single reflow. If the PCBA requires multiple reflows, place the module on the PCB during the final.
uProcessor Pinout
MAIN BOARD
Title: uPROCESSOR-ESP12E pinout
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 4 of 18


uPROCESSOR
Recommended a 100 µF electrolytic capacitor to filtrate current peaks. Also, we will introduce a ceramic capacitor $C_e = \frac{I}{\Delta V \cdot f}$. The I = 12 mA maximum, the ΔV= 3.3 V and the clock frequency is 80 MHz so our capacitor should be a 45.45 pF ceramic one. We will use a 22 pF as it is the most proximate value, we have available in the lab stock.
To ensure the power supply to the ESP32 chip during power-up, it is advised to add an RC delay circuit at the EN pin. The recommended setting for the RC delay circuit is usually R = 10 kΩ and C = 1 µF.
U5
ESP32-WROOM-32D
Pins SCK/CLK, SDO/SDO0, SDI/SD1, SDH/SD2 , SWP/SD3 , CSC/CMD are connected to the SPI FLASH so they shouldn't be used. As they correspond to GPIOs in the range 6-11 which are connected to the module's integrated SPI flash and SRAM.
For entering engineering mode, S4 and S5 should be pushed.
GPIOs 34 to 39 are input only pins. They don't have any pull-ups or pull-downs resistor. As they are configured as an ADC channel, we will simply disable the digital input buffer to prevent the possible problems.
For entering Booting mode: GPIO0 -> LOW, GPIO2 -> LOW, GPIO15 -> HIGH (unconnected), GPIO12 -> LOW
REAL MODEL
MAIN BOARD
Title: uPROCESSOR-ESP32
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 6 of 18

DATASHEET

	ESP-32-WROMM-32D
CPU	Xtensa LX6 32 bits Dual core from 80 MHz to 240 MHz
SRAM	520 kB (8 kB SRAM in RTC)
FLASH	4 MB (larger available)
ROM	448 kB
POWER SUPPLY	3.0 to 3.6 V
CURRENT CONSUMPTION	Minimum current required to be delivered: 500 mA Average Current Consumption:80mA Sleep: 2.5 µA (10 µA RTC + RTC memory) Low Consumption: >150 µA
TEMPERATURE RANGE	-40 to 85 ºC
Wi-Fi	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 150 Mbps
Wi-Fi MODES	Station / SOFTAP / SOFTAP + STATION / P2P
NETWROK PROTOCOL	Ipv4 / Ipv6 / SSL / TCP / UDP / HTTP / FTP / MQTT
BLUETOOTH	V4.2 BR/EDR+BLE
HW ENCRYPTATION	Yes

GPIO	38
DAC	2 DAC Channels 8 bit
ADC	SAR 12 bits 8 Channels
INTERFACES	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / DAC / TOUCH / PWM / LED/ SD card
TOUCH SENSOR	Yes (8 Channels)
TEMPERATURE SENSOR	Yes
HALL EFFECT SENSOR	Yes
DIMMENSIONS	18 x 25.5 x 3.1 mm^3 8 g
PRICE	2.20 € https://www.mouser.es/

The absolute maximum current drawn per GPIO is 40 mA.

GPIO0/GPIO2/GPIO4/GPIO5/GPIO12/GPIO15 can present an anormal behaviour during the reset. Use the Boot Mode Selection to make these pins work as expected after the resetting.

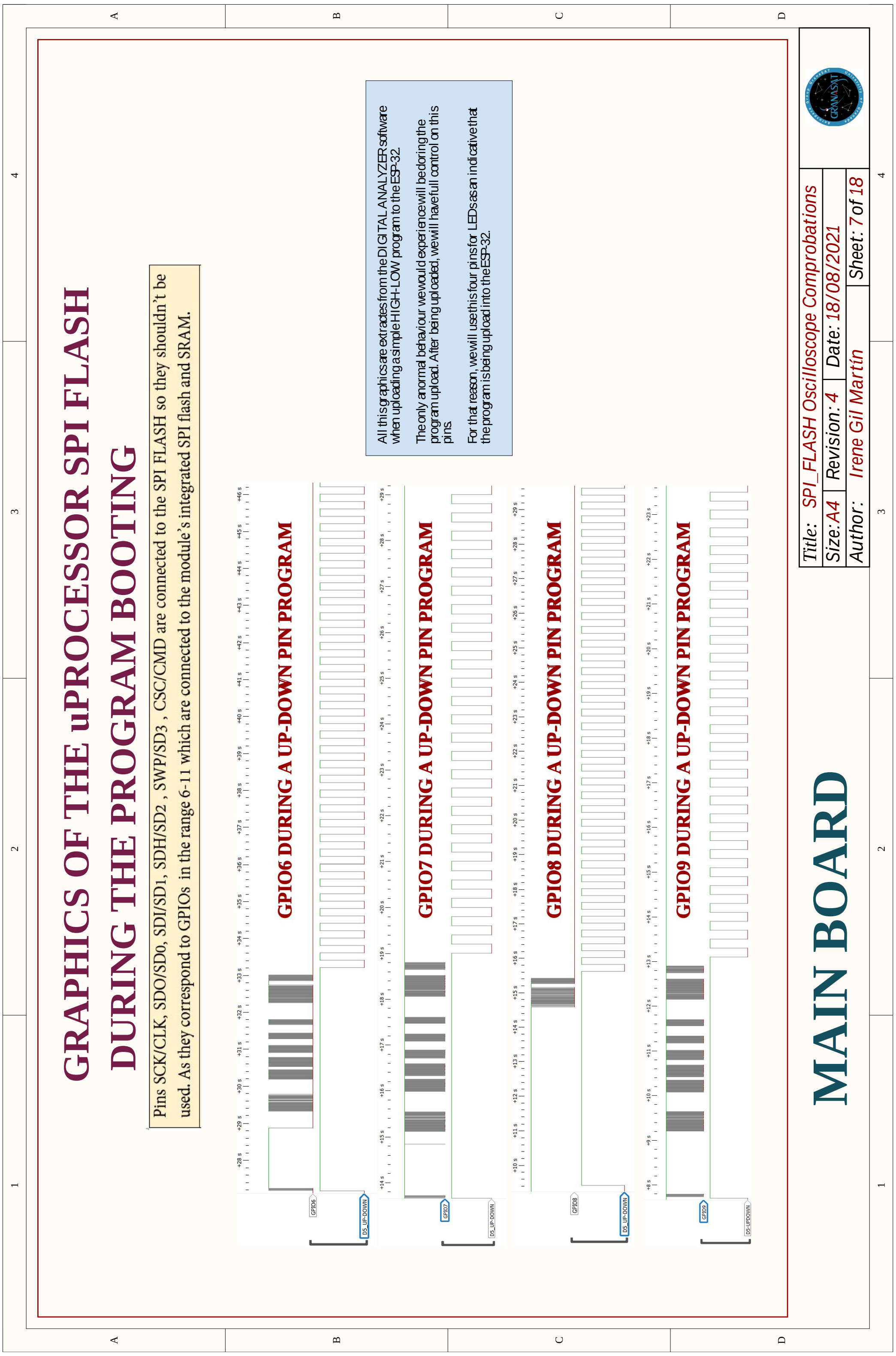
GPIOs 34 to 39 are input only pins. They don't have any pull-ups or pull-downs resistor. As they are configured as an ADC channel, we will simply disable the digital input buffer to prevent the possible problems.

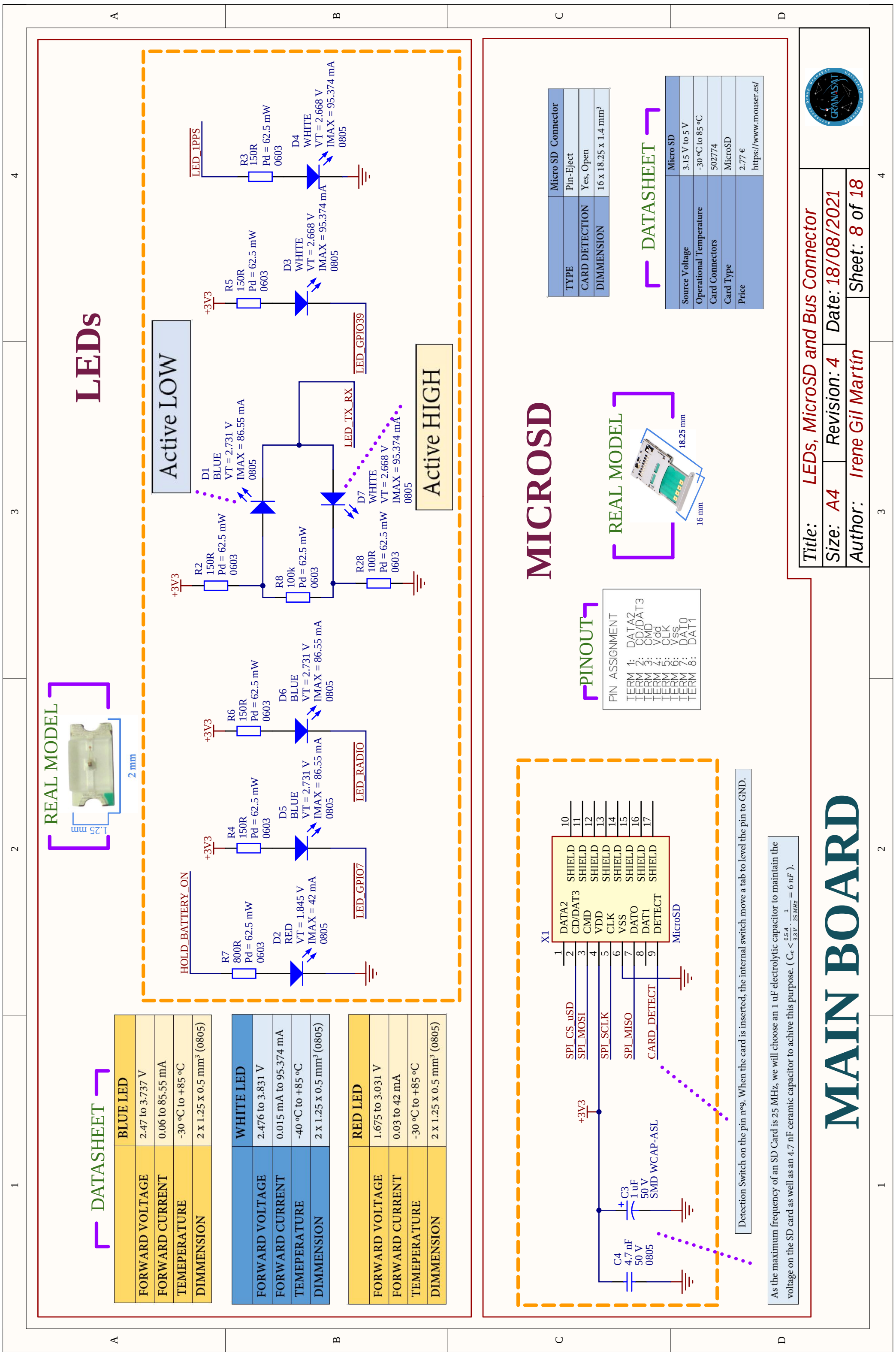
Solder the module in a single reflow. If the PCBA requires multiple reflows, place the module on the PCB during the final.

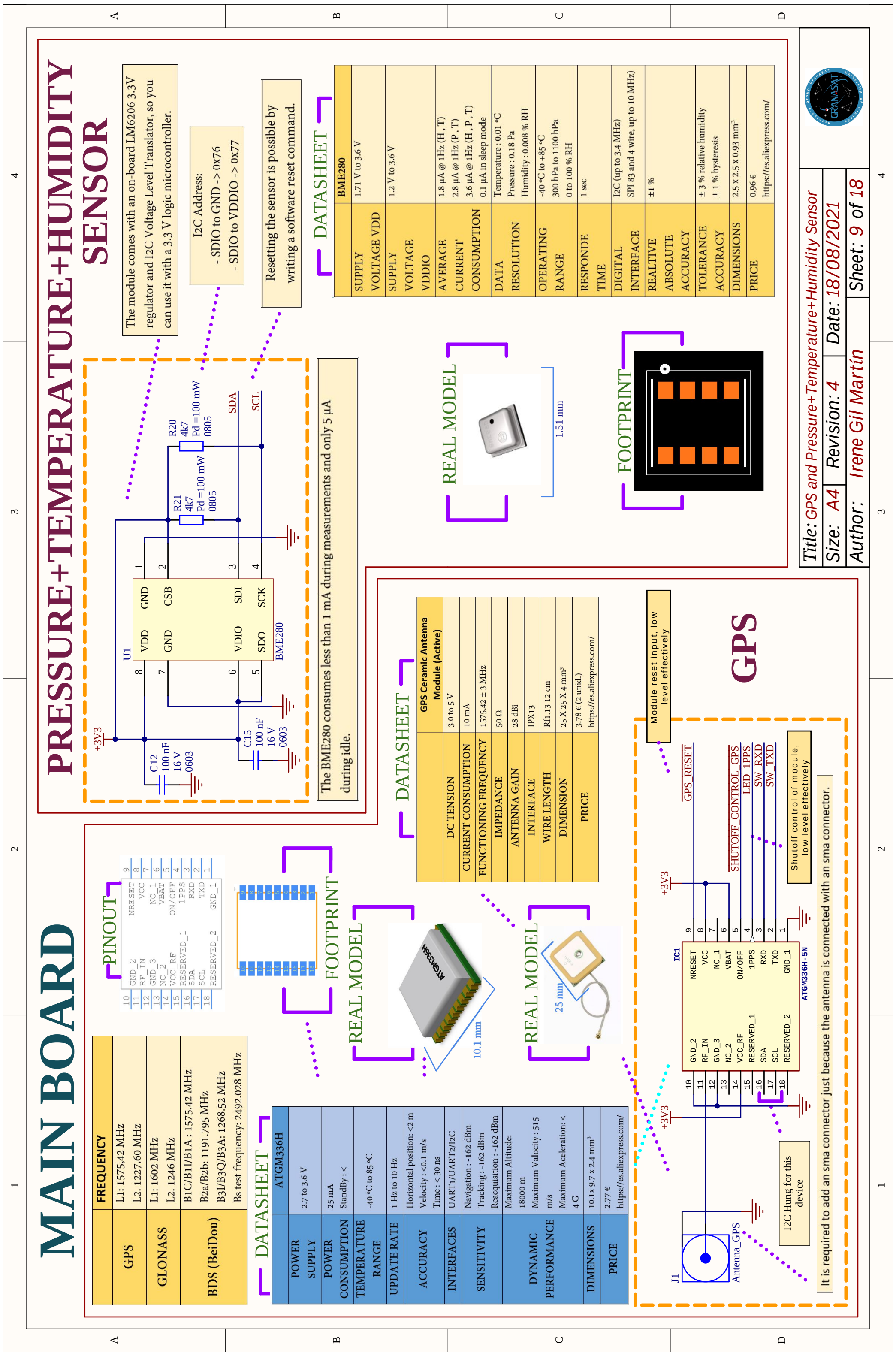
uPROCESSOR

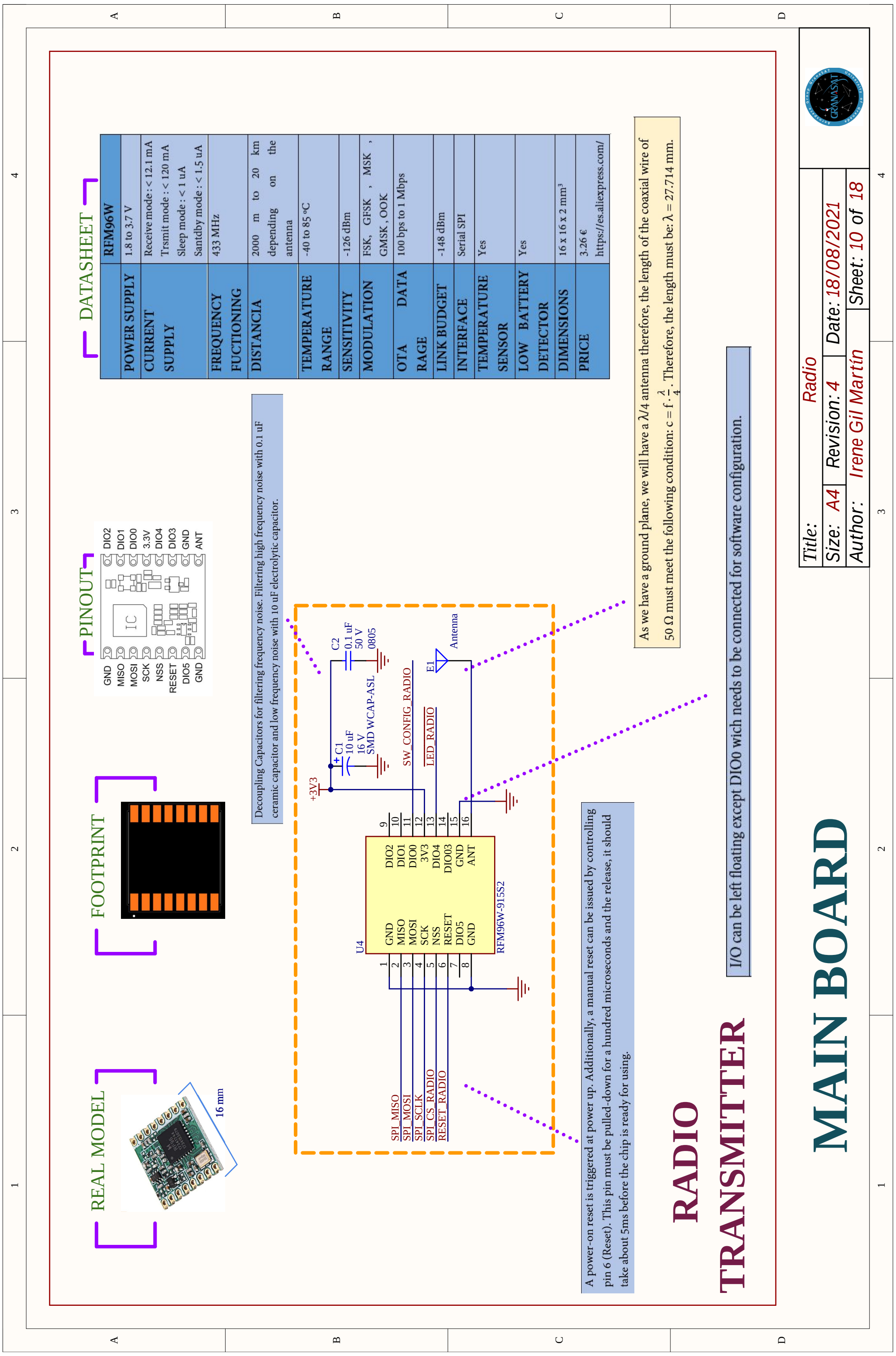
MAIN BOARD

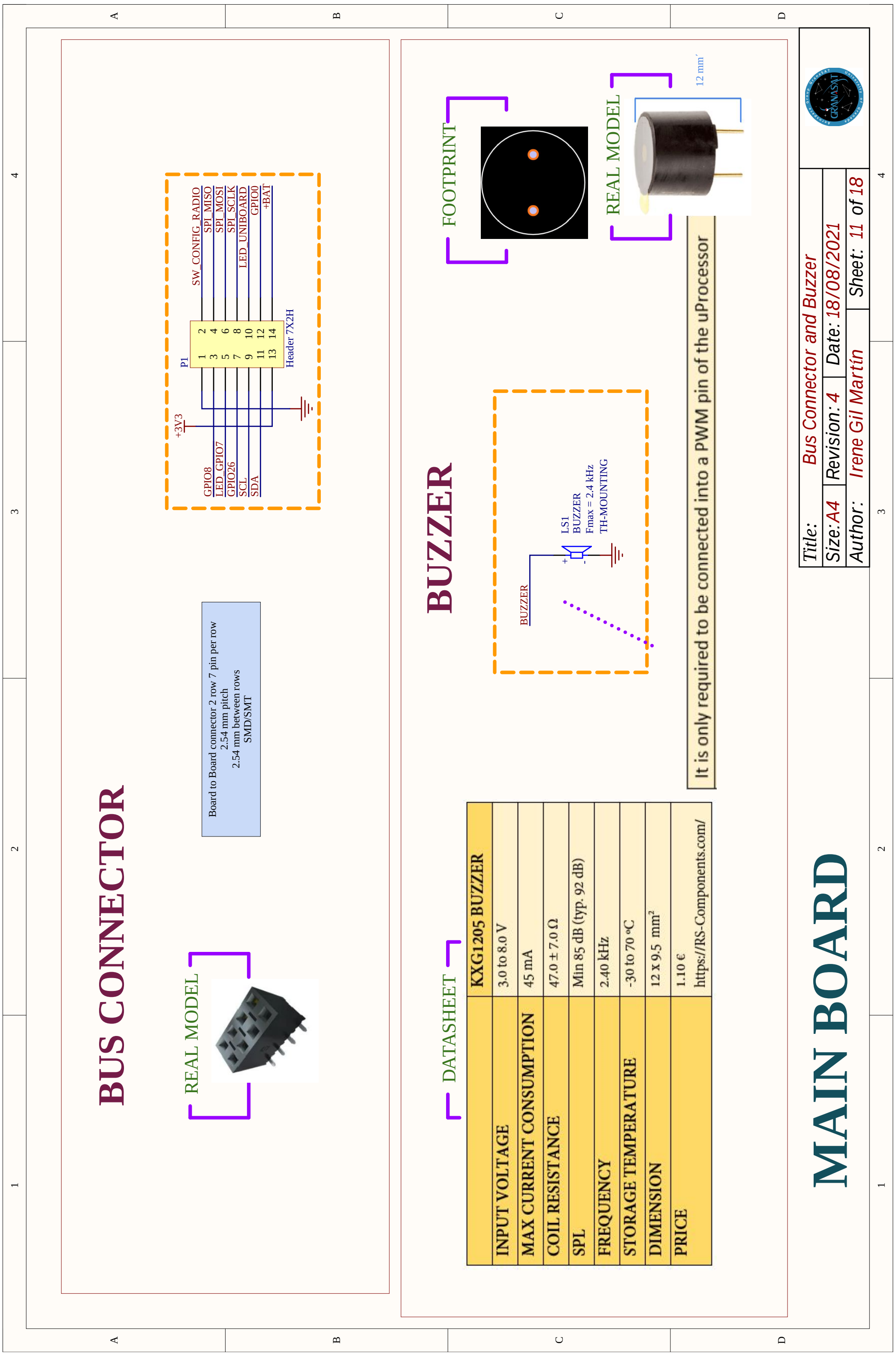
Title: uPROCESSOR-ESP32 information		
Size: A4	Revision: 4	Date: 18/08/2021
Author: Irene Gil Martín		Sheet: 5 of 18

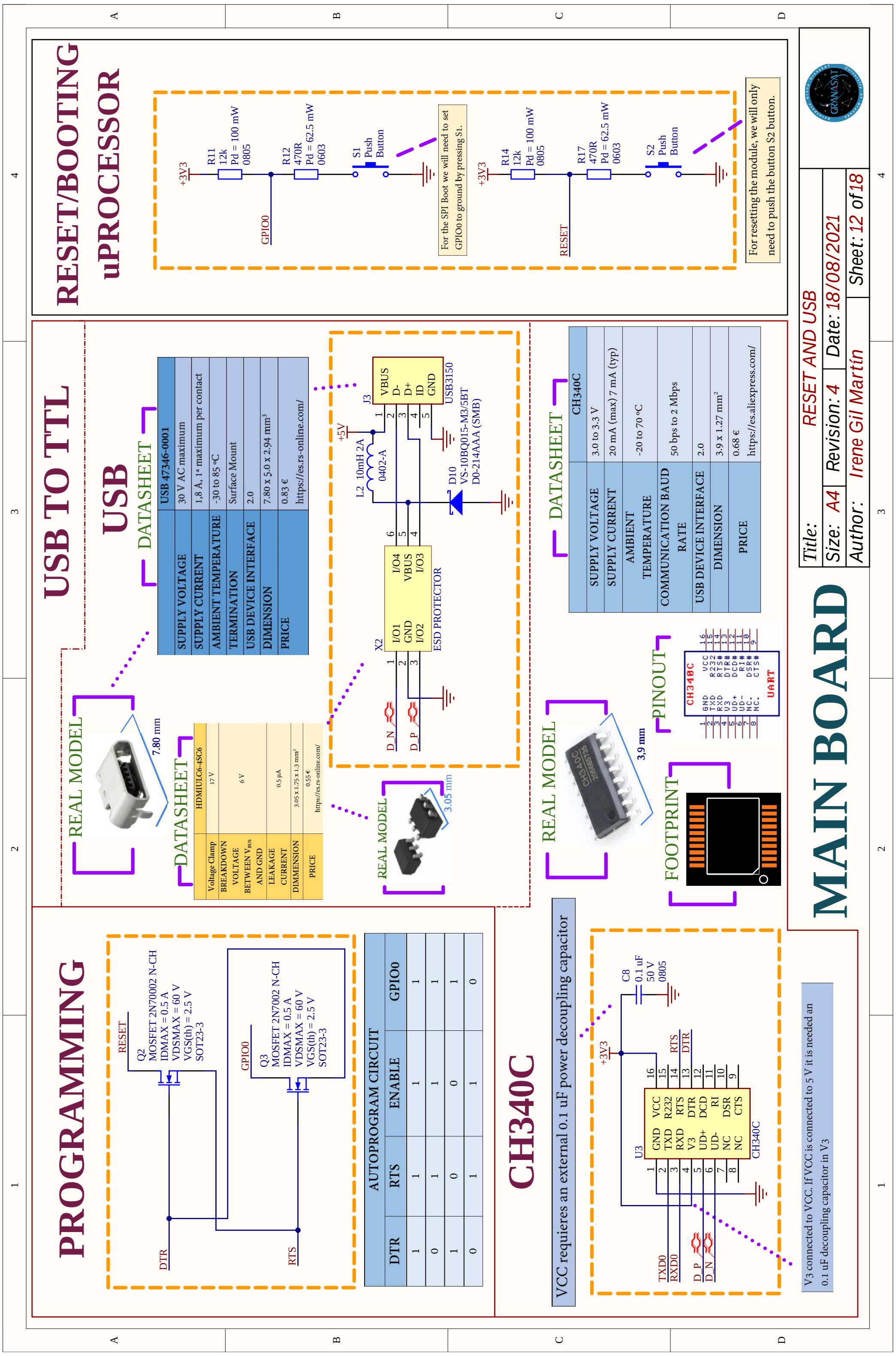

GRAPHICS OF THE uPROCESSOR SPI FLASH DURING THE PROGRAM BOOTING
Pins SCK/CLK, SDO/SD0, SDI/SD1, SDH/SD2 , SWP/SD3 , CSC/CMD are connected to the SPI FLASH so they shouldn't be used. As they correspond to GPIOs in the range 6-11 which are connected to the module's integrated SPI flash and SRAM.
All this graphics are extractes from the DIGITAL ANALYZER software when uploading a simple HIGH-LOW program to the ESP-32
The only anormal behaviour we would experience will be doring the program upload. After being uploaded, we will have full control on this pins.
For that reason, we will use this four pins for LEDs as an indicative that the program is being upload into the ESP-32.
GPIO6 DURING A UP-DOWN PIN PROGRAM
GPIO7 DURING A UP-DOWN PIN PROGRAM
GPIO8 DURING A UP-DOWN PIN PROGRAM
GPIO9 DURING A UP-DOWN PIN PROGRAM
MAIN BOARD
Title: SPI_FLASH Oscilloscope Comprobations
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 7 of 18

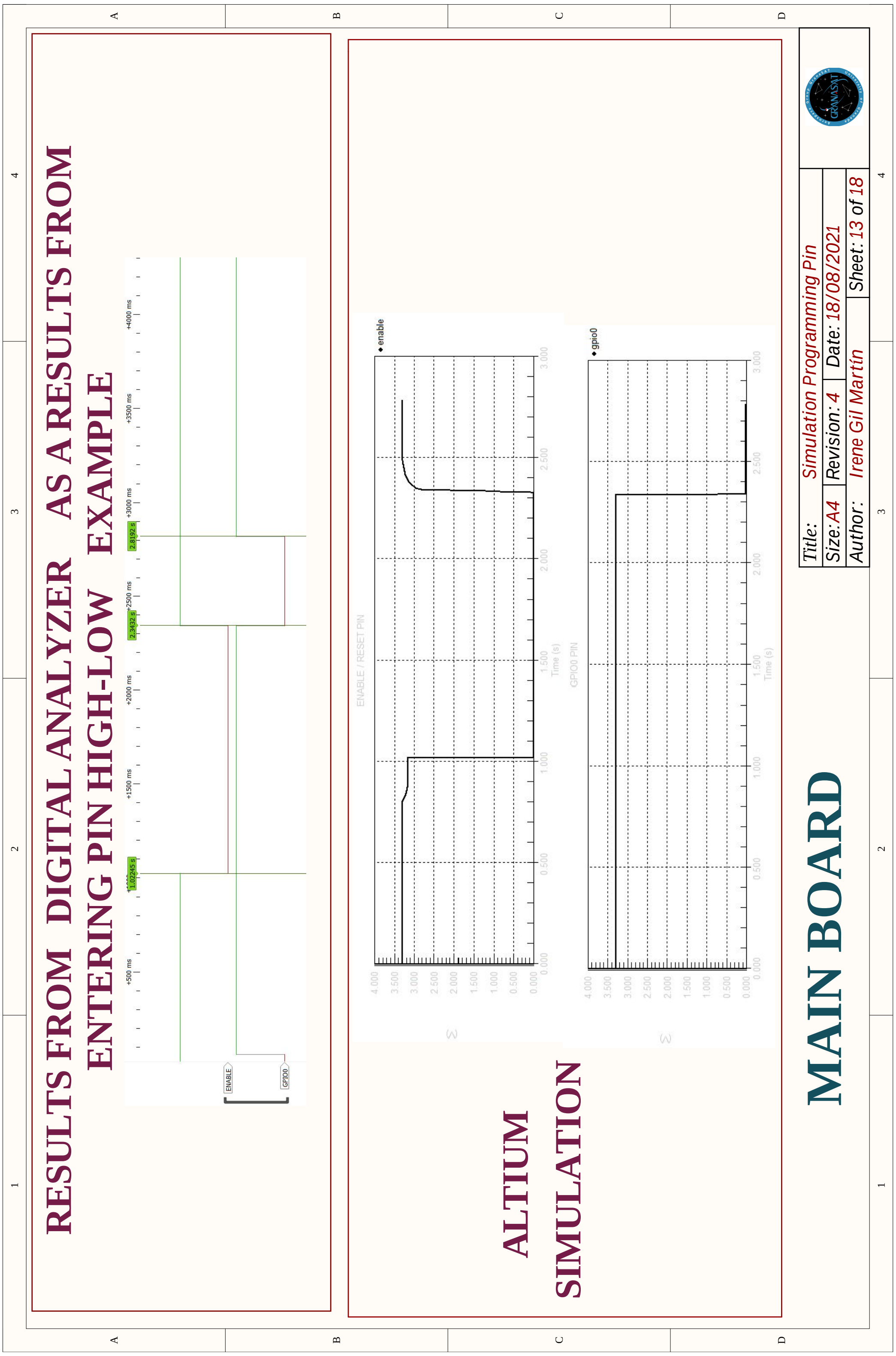

LEDs
Active LOW
Active HIGH
REAL MODEL
DATASHEET
BLUE LED
FORWARD VOLTAGE 2.47 to 3.737 V
FORWARD CURRENT 0.06 to 85.55 mA
TEMEPERATURE -30 ºC to +85 ºC
DIMMENSION 2 x 1.25 x 0.5 mm³ (0805)
WHITE LED
FORWARD VOLTAGE 2.476 to 3.831 V
FORWARD CURRENT 0.015 mA to 95.374 mA
TEMEPERATURE -40 ºC to +85 ºC
DIMMENSION 2 x 1.25 x 0.5 mm³ (0805)
RED LED
FORWARD VOLTAGE 1.675 to 3.031 V
FORWARD CURRENT 0.03 to 42 mA
TEMEPERATURE -30 ºC to +85 ºC
DIMMENSION 2 x 1.25 x 0.5 mm³ (0805)
MICROSD
PINOUT
Micro SD Connector
TYPE Pin-Eject
CARD DETECTION Yes, Open
DIMMENSION 16 x 18.25 x 1.4 mm³
Micro SD
Source Voltage 3.15 V to 5 V
Operational Temperature -30 ºC to 85 ºC
Card Connectors 502774
Card Type MicroSD
Price 2.77 €
https://www.mouser.es/
Detection Switch on the pin nº9. When the card is inserted, the internal switch move a tab to level the pin to GND.
MAIN BOARD
Title: LEDs, MicroSD and Bus Connector
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 8 of 18

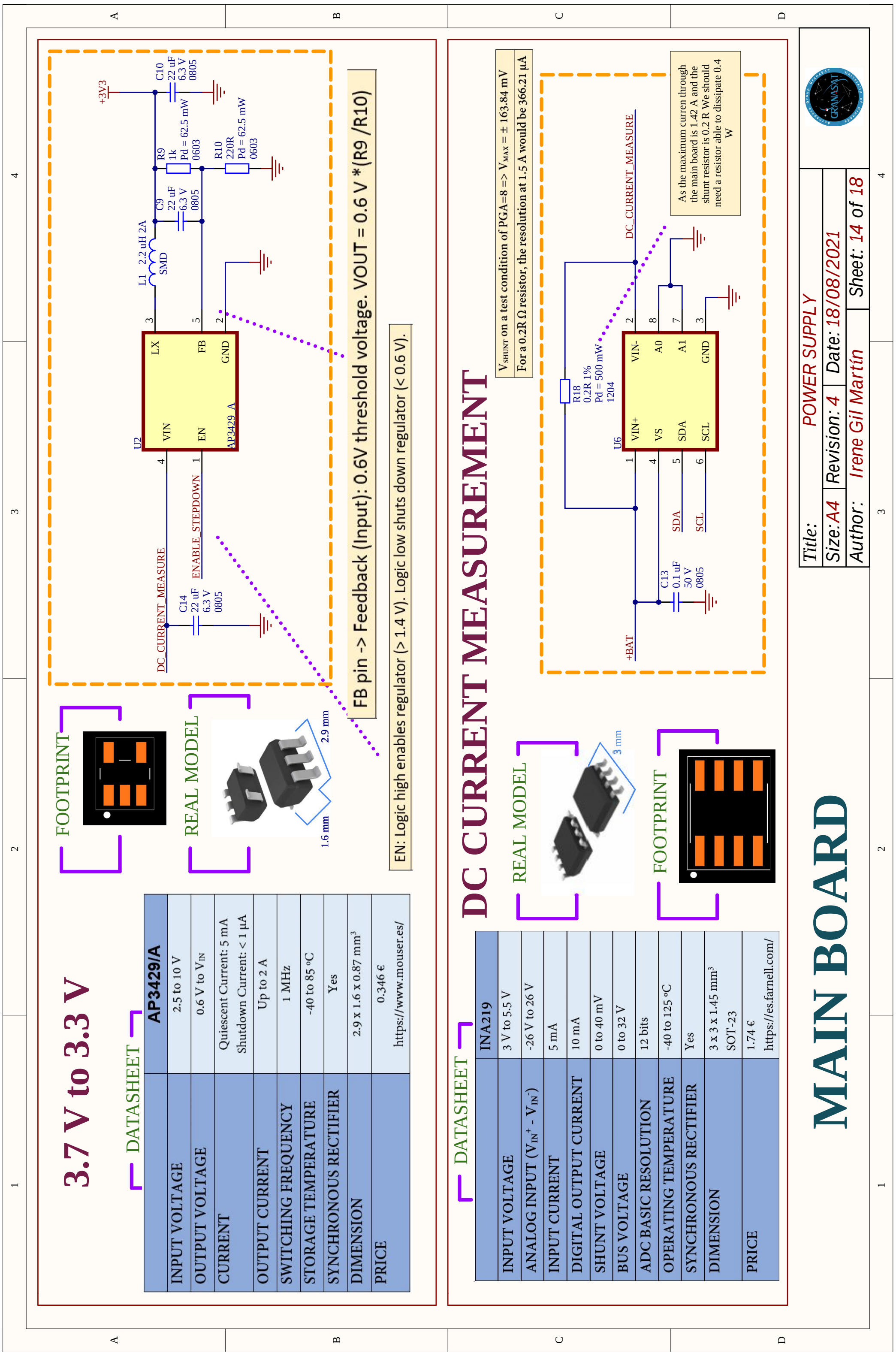

MAIN BOARD
PRESSURE+TEMPERATURE+HUMIDITY SENSOR
GPS
DATASHEET
PINOUT
FOOTPRINT
REAL MODEL
The module comes with an on-board LM6206 3.3V regulator and I2C Voltage Level Translator, so you can use it with a 3.3 V logic microcontroller.
I2C Address: - SDIO to GND -> 0x76 - SDIO to VDDIO -> 0x77
Resetting the sensor is possible by writing a software reset command.
The BME280 consumes less than 1 mA during measurements and only 5 μA during idle.
It is required to add an sma connector just because the antenna is connected with an sma connector.
Module reset input, low level effectively
Shutoff control of module, low level effectively
I2C Hung for this device
Title: GPS and Pressure+Temperature+Humidity Sensor
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 9 of 18

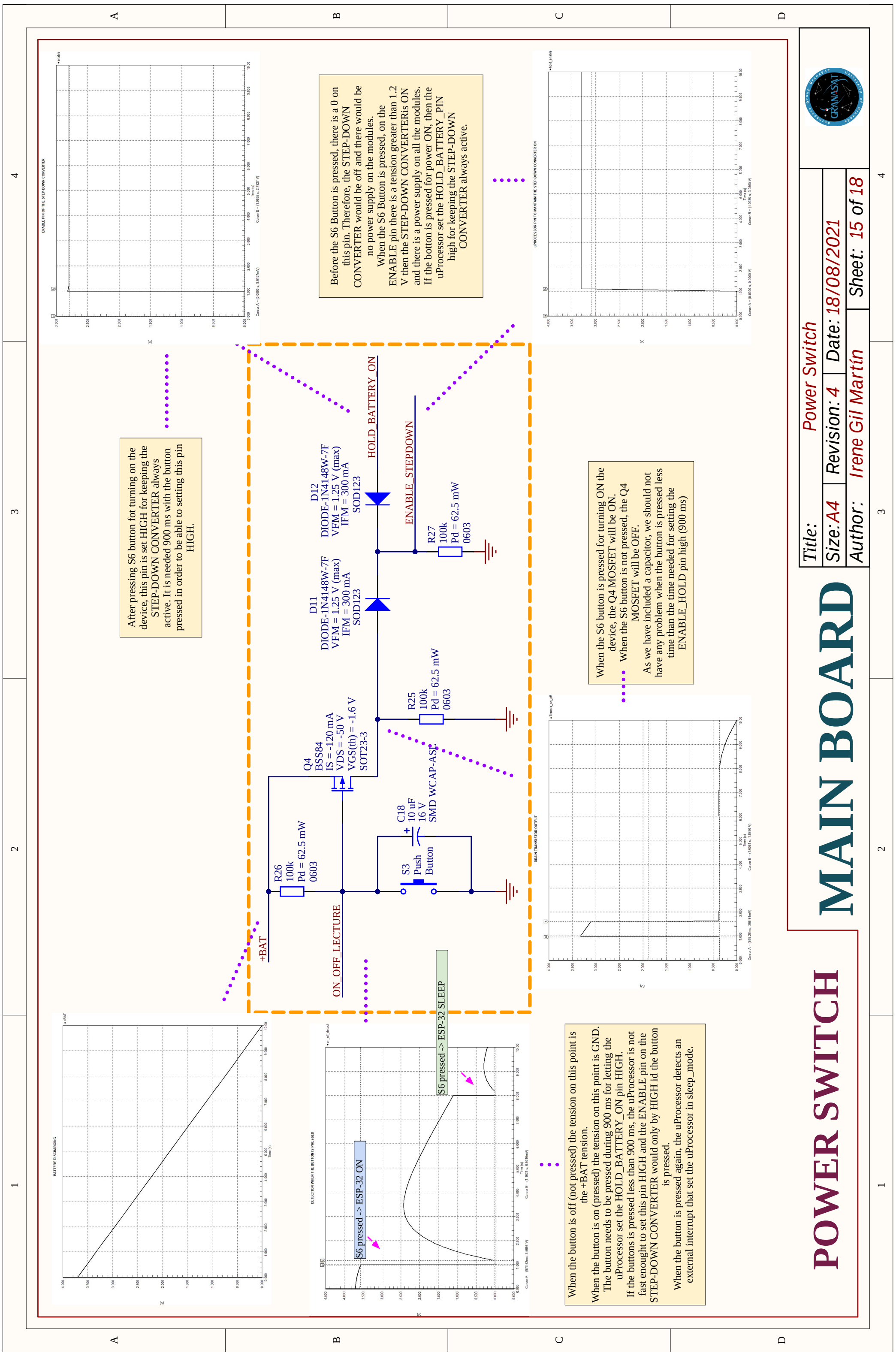

REAL MODEL
16 mm
FOOTPRINT
PINOUT
GND MISO MOSI SCK NSS RESET DIO5 GND
DIO2 DIO1 DIO0 3.3V DIO4 DIO3 GND ANT
DATASHEET
RFM96W
POWER SUPPLY 1.8 to 3.7 V
CURRENT SUPPLY Receive mode : < 12.1 mA Trsmit mode : < 120 mA Sleep mode : < 1 uA Santdby mode : < 1.5 uA
FREQUENCY FUCTIONING 433 MHz
DISTANCIA 2000 m to 20 km depending on the antenna
TEMPERATURE RANGE -40 to 85 ºC
SENSITIVITY -126 dBm
MODULATION FSK, GFSK, MSK, GMSK, OOK
OTA DATA RAGE 100 bps to 1 Mbps
LINK BUDGET -148 dBm
INTERFACE Serial SPI
TEMPERATURE SENSOR Yes
LOW BATTERY DETECTOR Yes
DIMENSIONS 16 x 16 x 2 mm³
PRICE 3.26 € https://es.aliexpress.com/
Decoupling Capacitors for filtering frequency noise. Filtering high frequency noise with 0.1 uF ceramic capacitor and low frequency noise with 10 uF electrolytic capacitor.
As we have a ground plane, we will have a λ/4 antenna therefore, the length of the coaxial wire of 50 Ω must meet the following condition: c = f · λ/4. Therefore, the length must be: λ = 27.714 mm.
+3V3
C1 10 uF 16 V SMD WCAP-ASL
C2 0.1 uF 50 V 0805
SW_CONFIG_RADIO
LED_RADIO
E1 Antenna
U4
RFM96W-915S2
SPI_MISO
SPI_MOSI
SPI_SCLK
SPI_CS_RADIO
RESET_RADIO
A power-on reset is triggered at power up. Additionally, a manual reset can be issued by controlling pin 6 (Reset). This pin must be pulled-down for a hundred microseconds and the release, it should take about 5ms before the chip is ready for using.
I/O can be left floating except DIO0 wich needs to be connected for software configuration.
RADIO TRANSMITTER
MAIN BOARD
Title: Radio
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martín Sheet: 10 of 18


BUS CONNECTOR
REAL MODEL
Board to Board connector 2 row 7 pin per row
2.54 mm pitch
2.54 mm between rows
SMD/SMT
P1
Header 7X2H
+3V3
GPIO8
LED_GPIO7
GPIO26
SCL
SDA
SW_CONFIG_RADIO
SPI_MISO
SPI_MOSI
SPI_SCLK
LED_UNIBOARD
GPIO0
+BAT
BUZZER
LS1
BUZZER
Fmax = 2.4 kHz
TH-MOUNTING
FOOTPRINT
REAL MODEL
12 mm
It is only required to be connected into a PWM pin of the uProcessor
DATASHEET
KXG1205 BUZZER
INPUT VOLTAGE
3.0 to 8.0 V
MAX CURRENT CONSUMPTION
45 mA
COIL RESISTANCE
47.0 ± 7.0 Ω
SPL
Min 85 dB (typ. 92 dB)
FREQUENCY
2.40 kHz
STORAGE TEMPERATURE
-30 to 70 ºC
DIMENSION
12 x 9.5 mm²
PRICE
1.10 €
https://RS-Components.com/
MAIN BOARD
Title: Bus Connector and Buzzer
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 11 of 18


RESET/BOOTING uPROCESSOR
For the SPI Boot we will need to set GPIO0 to ground by pressing S1.
For resetting the module, we will only need to push the button S2 button.
USB TO TTL
USB
DATASHEET
REAL MODEL
ESD PROTECTOR
PROGRAMMING
AUTOPROGRAM CIRCUIT
CH340C
VCC requieres an external 0.1 uF power decoupling capacitor
V3 connected to VCC. If VCC is connected to 5 V it is needed an 0.1 uF decoupling capacitor in V3
PINOUT
FOOTPRINT
MAIN BOARD
Title: RESET AND USB
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 12 of 18


RESULTS FROM DIGITAL ANALYZER AS A RESULTS FROM ENTERING PIN HIGH-LOW EXAMPLE
ALTIUM SIMULATION
MAIN BOARD
Title: Simulation Programming Pin
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 13 of 18


3.7 V to 3.3 V
DATASHEET
AP3429/A
INPUT VOLTAGE 2.5 to 10 V
OUTPUT VOLTAGE 0.6 V to VIN
CURRENT Quiescent Current: 5 mA Shutdown Current: < 1 µA
OUTPUT CURRENT Up to 2 A
SWITCHING FREQUENCY 1 MHz
STORAGE TEMPERATURE -40 to 85 ºC
SYNCHRONOUS RECTIFIER Yes
DIMENSION 2.9 x 1.6 x 0.87 mm³
PRICE 0.346 €
https://www.mouser.es/
FOOTPRINT
REAL MODEL
2.9 mm
1.6 mm
FB pin -> Feedback (Input): 0.6V threshold voltage. VOUT = 0.6 V *(R9 /R10)
EN: Logic high enables regulator (> 1.4 V). Logic low shuts down regulator (< 0.6 V).
U2
AP3429_A
VIN
EN
LX
FB
GND
+3V3
L1 2.2 uH 2A SMD
C9 22 uF 6.3 V 0805
C10 22 uF 6.3 V 0805
R9 1k Pd = 62.5 mW 0603
R10 220R Pd = 62.5 mW 0603
C14 22 uF 6.3 V 0805
DC_CURRENT_MEASURE
ENABLE_STEPDOWN
DC CURRENT MEASUREMENT
DATASHEET
INA219
INPUT VOLTAGE 3 V to 5.5 V
ANALOG INPUT (VIN+ - VIN-) -26 V to 26 V
INPUT CURRENT 5 mA
DIGITAL OUTPUT CURRENT 10 mA
SHUNT VOLTAGE 0 to 40 mV
BUS VOLTAGE 0 to 32 V
ADC BASIC RESOLUTION 12 bits
OPERATING TEMPERATURE -40 to 125 ºC
SYNCHRONOUS RECTIFIER Yes
DIMENSION 3 x 3 x 1.45 mm³ SOT-23
PRICE 1.74 €
https://es.farnell.com/
REAL MODEL
3 mm
FOOTPRINT
VSHUNT on a test condition of PGA=8 => VMAX = ± 163.84 mV
For a 0.2R Ω resistor, the resolution at 1.5 A would be 366.21 µA
As the maximum curren through the main board is 1.42 A and the shunt resistor is 0.2 R We should need a resistor able to dissipate 0.4 W
U6
VIN+
VIN-
VS
A0
SDA
A1
SCL
GND
R18 0.2R 1% Pd = 500 mW 1204
C13 0.1 uF 50 V 0805
+BAT
SDA
SCL
DC_CURRENT_MEASURE
MAIN BOARD
Title: POWER SUPPLY
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martín Sheet: 14 of 18
GRANASAT


POWER SWITCH
MAIN BOARD
Title: Power Switch
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 15 of 18
Before the S6 Button is pressed, there is a 0 on this pin. Therefore, the STEP-DOWN CONVERTER would be off and there would be no power supply on the modules. When the S6 Button is pressed, on the ENABLE pin there is a tension greater than 1.2 V then the STEP-DOWN CONVERTERis ON and there is a power supply on all the modules. If the botton is pressed for power ON, then the uProcessor set the HOLD_BATTERY_PIN high for keeping the STEP-DOWN CONVERTER always active.
After pressing S6 button fot turning on the device, this pin is set HIGH for keeping the STEP-DOWN CONVERTER always active. It is needed 900 ms with the button pressed in order to be able to setting this pin HIGH.
When the S6 button is pressed for turning ON the device, the Q4 MOSFET will be ON. When the S6 button is not pressed, the Q4 MOSFET will be OFF. As we have included a capacitor, we should not have any problem when the button is pressed less time than the time needed for setting the ENABLE_HOLD pin high (900 ms)
When the button is off (not pressed) the tension on this point is the +BAT tension. When the button is on (pressed) the tension on this point is GND. The button needs to be pressed during 900 ms for letting the uProcessor set the HOLD_BATTERY_ON pin HIGH. If the buttons is pressed less than 900 ms, the uProcessor is not fast enought to set this pin HIGH and the ENABLE pin on the STEP-DOWN CONVERTER would only by HIGH id the button is pressed. When the button is pressed again, the uProcessor detects an external interrupt that set the uProcessor in sleep_mode.
HOLD_BATTERY_ON
ENABLE_STEPDOWN
+BAT
ON_OFF_LECTURE
D12 DIODE-1N4148W-7F VFM = 1.25 V (max) IFM = 300 mA SOD123
D11 DIODE-1N4148W-7F VFM = 1.25 V (max) IFM = 300 mA SOD123
R27 100k Pd = 62.5 mW 0603
R25 100k Pd = 62.5 mW 0603
R26 100k Pd = 62.5 mW 0603
Q4 BSS84 IS = -120 mA VDS = -50 V VGS(th) = -1.6 V SOT23-3
C18 10 uF 16 V SMD WCAP-ASL
S3 Push Button
S6 pressed -> ESP-32 ON
S6 pressed -> ESP-32 SLEEP

DEVICES TRADE-OFF

uPROCESSOR

	SAMD21G18A-AUT	ESP-32	ESP-12
CPU	ARM CORTEX -M0+CPU running at up to 48 MHz: - Single-cycle hardware multiplier - Micro Trace Buffer (MTB)	Xtensa LX6 32 bits Single/Dual core Up to 600 MIPS	Tensilica L X106 32 bits Single core
SRAM	32 kB	520 kB (16 kB SRAM in RTC)	< 50 kB
FLASH	256	4 MB (larger available)	4 MB
ROM	No	448 kB	No Porgrammable ROM
POWER SUPPLY	1.62 to 3.63 V	2.3 to 3.6 V	3.0 to 3.6 v
CURRENT CONSUMPTION	85 to 1.2 mA	Max: 225 mA Typ:80mA Sleep: 2.5 µA (10 µA RTC + memoria RTC) Low Consumption: >150 µA	Max: 225 mA Typ:80mA Sleep: 20 µA (RTC + memoria RTC)
TEMPERATURE RANGE	-40 to 125 ºC	-40 to 125 ºC	-40 to 125 ºC
Wi-Fi	No	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 150 Mbps	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 72.2Mbps
Wi-Fi MODES	-	Station / SOFTAP / SOFTAP + STATION / P2P	Station / SOFTAP / SOFTAP + STATION / P2P
NETWROK PROTOCOL	-	Ipv4 / Ipv6 / SSL / TCP / UDP / HTTP / FTP / MQTT	Ipv4 / TCP / UDP / HTTP / MQTT
BLUETOOTH	No	V4.2 BR/EDR+BLE	No
ETHERNET MAC INTERFACE	-	10/100 Mbps	No

	SAMD21G18A-AUT	ESP-32	ESP-12
HW ENCRYPTATION	No	Yes	No (TLS 1.2 in SW)
GPIO	Up to 52	36	17
DAC	10 bit	2 DAC channels 8 bit	NO
ADC	12 bit Up to 20 Channels	SAR 12 bits 8 Channels	SAR 10 bits 1 Channel
INTERFACES	DMAC / TC / TCC / RTC / WDT / CRC / SERCOM / USART / 12C / SPI / ADC / DAC / AC / PTC	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / DAC / TOUCH / PWM / LED	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / PWM / LED
TOUCH SENSOR	Yes	Yes (8 Channels)	No
TEMPERATURE SENSOR	No	Yes	No
HALL EFFECT SENSOR	No	Yes	No
DIMMENSIONS	7 x 7 x 1.0 mm³	57 x 28 x 1 mm³ 8 g	16 x 24 x 3.4 mm³ 1.72 g
PRICE	2.73 €	2.20 € https://www.mouser.es/	3 – 6€
PREFERENCE ORDER	3	1	2
ELECTION	✗	✓	✗

Title: uProcessor's Trade-off		
Size: A4	Revision: 2	Date: 18/08/2021
Author: Irene Gil Martín		Sheet: 16 of 18

DEVICES TRADE-OFF

GPS

	Ublox-MAX8C	ATGM336H	CAM-M8C	BG01-H111S100
POWER SUPPLY	1.65 – 3.6 V	2.7 to 3.6 V	1.65 TO 3.6 V	Typ. 3.3 V
POWER CONSUMPTION	Low power consumption	25 mA StandBy : <	71 mA	Acquisition Current: 30 mA Tracking Current: 20 mA
TEMPERATURE RANGE	-40 ºC to 85 ºC	-40 ºC to 85 ºC	-40 ºC to 85 ºC	-40 ºC to 85 ºC
UPDATE RATE	0.25 Hz to 10 MHz	1 Hz to 10 Hz	10 Hz	
ACCURACY	Horizontal position <4 m Velocity : <0.05 m/s Time:60 ns	Horizontal position: <2 m Velocity : <0.1 m/s Time : < 30 ns	Horizontal position: Velocity : <0.05 m/s Acceleration:	Horizontal position: 2.5 m Altitude position: 3.5 m Velocity : 0.1 m/s Time pulse signal: 30 ns
INTERFACES	UART , DDC (I2C fast mode compatible)	UART1/UART2/I2C	UART / DDC / I2C	UART
SENSITIVITY	Navigation : -166 dBm Tracking : -166 dBm Reacquisition : -160 dBm	Navigation : -162 dBm Tracking : -162 dBm Reacquisition : -162 dBm	Navigation : -164 dBm Tracking : -164 dBm Reacquisition : -160 dBm	Cold Start : -148 dBm Warm Start: -162 dBm Tracking : -166 dBm Reacquisition : -164 dBm
DYNAMIC PERFORMANCE	Maximum Altitude: 50000 m Maximum Valocity : 500 m/s Maximum Aceleration: < 4 G	Maximum Altitude: 18000 m Maximum Valocity : 515 m/s Maximum Aceleration: < 4 G	Maximum Altitude: 50000 m Maximum Valocity : 500 m/s Maximum Aceleration: < 4 G	Cold Start: 27.5 s Warm Start: < 1 s Re-Acquisition: < 1 s A-GNSS: < 1 s
DIMENSIONS	9.7 x 10.1 mm²	10.1x 9.7 x 2.4 mm³ 0.6 g	9.60 x 14 x 2 mm³	16.2 x 12.2 mm²
PRICE	5.62 € https://es.aliexpress.com/	2.77 € https://es.aliexpress.com/		6.75 € www.mecter.com
PREFERENCE ORDER	4	1	3	2
ELECTION	✗	✓	✗	✗

RADIO TRANSMITTER

	RFM69W	SX1278	RFM96W	HM-TRP
POWER SUPPLY	2.4 to 3.6 Vdc	1.8 to 3.7 V	1.8 to 3.7 V	2.4 to 3.6 V
CURRENT SUPPLY	Receive mode : 16 mA Trsmit mode : 16 – 45 mA Sleep mode : < 1 uA Santdby mode : <1.5 mA	Receive mode : <12 mA Trsmit mode : < 20 mA Sleep mode : < 1 uA Santdby mode : <1.8 mA	Receive mode : < 12.1 mA Trsmit mode : < 120 mA Sleep mode : < 1 uA Santdby mode : < 1.5 uA	Receive mode : < 30 mA Trsmit mode : < 120 mA Sleep mode : 2 uA
FREQUENCY FUCTIONING	433 MHz	433 MHz	433 MHz	414 to 454 MHz ISM band
DISTANCIA	500 m	5000 m	2000 m to 20 km depending on the antenna	Over 1 km in open air
TEMPERATURE RANGE	- 20 to 70 ºC	-40 to 85 ºC	-40 to 85 ºC	-40 to 85 ºC
SENSITIVITY	-120 dBm	Down to -148 dBm	-126 dBm	-117 dBm
MODULATION	FSK , GFSK , MSK , GMSK or OOK	FSK, GFSK, MSK, GMSK, OOK	FSK, GFSK , MSK , GMSK , OOK	FSK 2 way half-duplex communication, strong anti-interfere
OTA DATA RAGE	1.2 to 300 kbps	1 to 300 kbps	100 bps to 1 Mbps	1.2 kbps to 115.2 kbps
LINK BUDGET	115 dB	168 dB	-148 dBm	
INTERFACE	Serial SPI	Serial SPI	Serial SPI	Standard TTL UART, extendable to RS232 or other interface
TEMPERATURE SENSOR	Yes	Yes	Yes	No
LOW BATTERY DETECTOR	No	Yes	Yes	Yes
DIMENSIONS	19.7 x 16 x 1.9 mm³	17.2 X 17 mm²	16 x 16 x 2 mm³	16 x 20 x 2 mm³
PRICE	1.52 € https://es.aliexpress.com/	3.13 € https://es.aliexpress.com/	3.26 € https://es.aliexpress.com/	7.25 € www.mecter.com
PREFERENCE ORDER	4	3	1	2
ELECTION	✗	✗	✓	✗

Title: GPS and Radio Trade-off
Size: A4 Revision: 2 Date: 18/08/2021
Author: Irene Gil Martín Sheet: 17 of 18

MAXIMUM CONSUMPTION BUDGET					
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT_MAX} (A)	I_{OUT} (A)
AP3428/A (3.7 V -> 3.3 V)	1	3,6	6,6	2	2
LM3671 (3.7 V -> 3.3 V)	2	3,6	3,795	1,15	1,02

	TYP	MAX
V_{SOURCE} (V)	3,7	3,7
CAPACITY (mAh)	900	900
SOURCE SPECS	3.7 V / 3.3 V	3.7 V / 3.3 V
TOTAL POWER MAIN BOARD(W)	2,31	6,65
TOTAL POWER BASE BOARD(W)	2,50	6,58
CURRENT NEEDED MAIN BOARD (A)	0,70	1,98
CURRENT NEEDED BASE BOARD (A)	0,76	1,98

TYPICAL CONSUMPTION BUDGET					
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT_MAX} (A)	I_{OUT} (A)
AP3429/A (3.7 V -> 3.3 V)	1	3,6	4,95	2	1,5
LM3671 (3.7 V -> 3.3 V)	2	3,6	1,98	0,8	0,6

Component	Module	Number/PCB	V_{IN} (V)	I_{IN_TYP} (mA)	I_{IN_MAX} (mA)	P_{IN_TYP} (W)	P_{IN_MAX} (W)
uProcessor	ESP-32-WROOM32E	1	3,3	500	1000	1,65	3,30
microSD	SanDisk standard uSD	1	3,3	30	100	0,10	0,33
GPS	ATGM-336H	1	3,3	100	100	0,33	0,33
	Ceramic Antenna	1	3,3	10	10	0,03	0,03
	Exterior Antenna	1	3,3	10	10	0,03	0,03
Radio Transmitter	RFM96W	1	3,3	20	120	0,07	0,40
Radio Receiver	RFM96W	1	3,3	10,8	12,1	0,04	0,04
Pressure	BME280	1	3,3	2,80E-03	4,20E-03	9,24E-06	1,39E-05
Humidity Sensor				1,80E-03	2,80E-03	5,94E-06	9,24E-06
Temperature				1,00E-03	2,00E-03	3,30E-06	3,60E-09
Buzzer	KXG1205	1	3,3	10	45	0,03	0,15
Display	ILI9341	1	3,3	80	100	0,26	0,33
USB2TTL	CH340C	1	3,3	7	20	0,02	0,14
Red LED	0805 SMD	1	3,3	1,56	42	0,01	0,14
Blue LED	0805SMD	3	3,3	5,40	258	0,02	0,85
White LED	0805 SMD	3	3,3	4,08	285	0,01	0,94
GREEN LED	0603 SMD	1	3,3	1,00	1,13	0,00	0,004
Red LED	0603 SMD	1	3,3	0,76	0,83	2,49E-03	2,74E-03
DC Current Measurement	INA219A	1	3,3	10	10	0,033	0,033

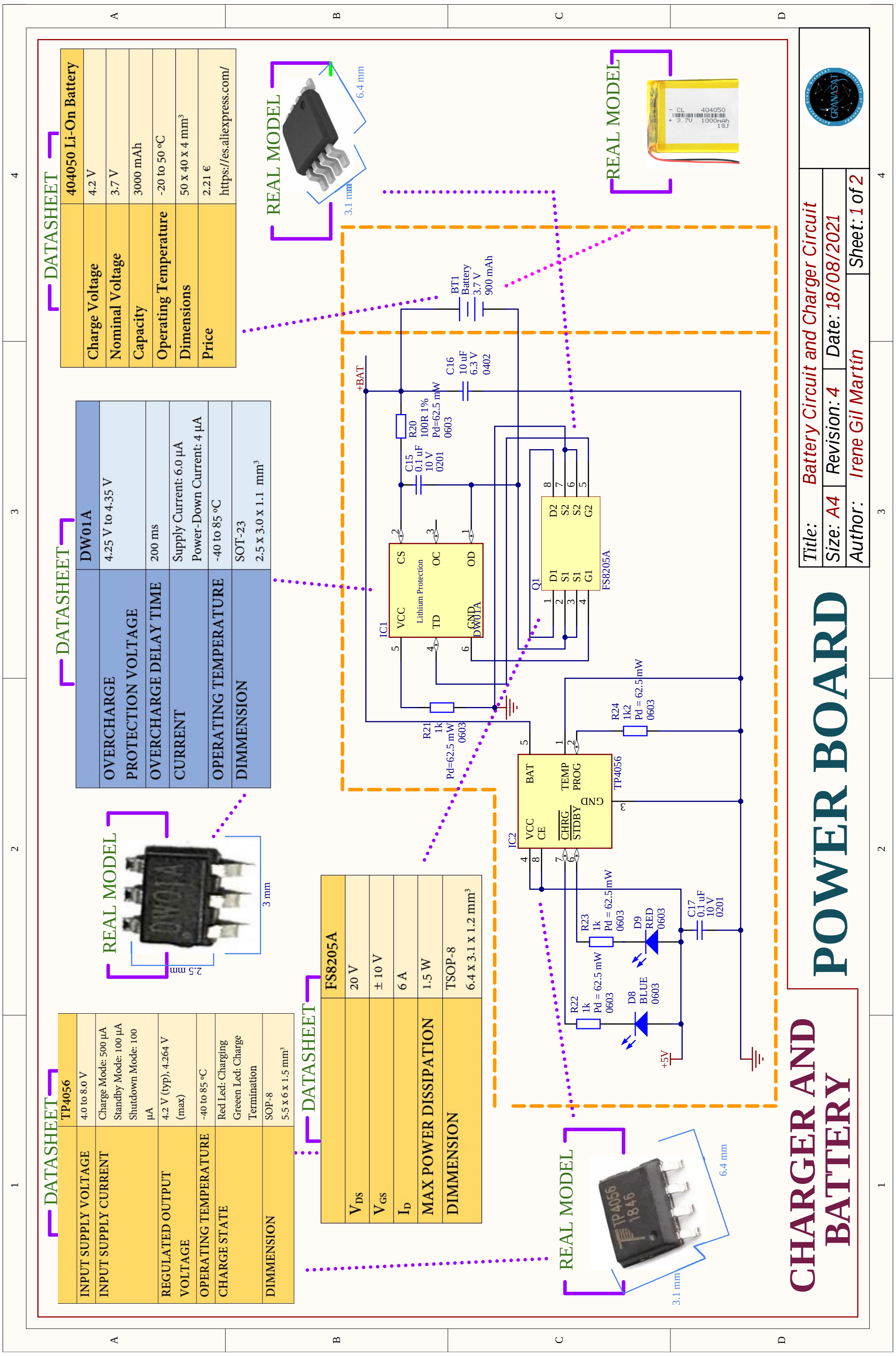
CURRENT CONSUMPTION	TYP (A)	MAX (A)
AP3429/A_CURRENT-CONSUMPTION_MB	0,80	0,02
LM3671_CURRENT-CONSUMPTION_MB	0,50	0,06
AP3429/A_CURRENTCONSUMPTION_BB	0,74	0,02
LM3671_CURRENT-CONSUMPTION_BB	0,44	0,06

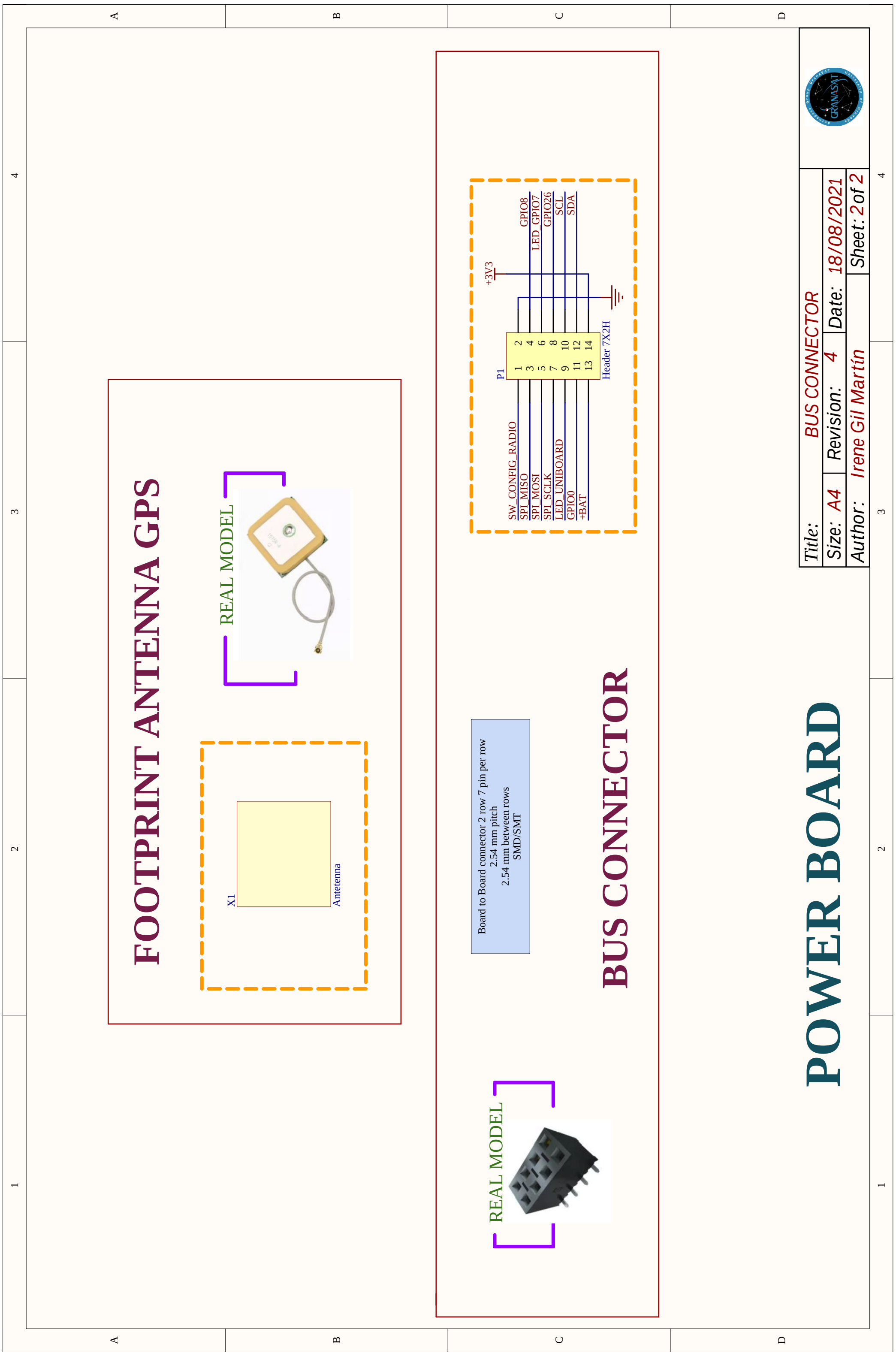
POWER CONSUMPTION	TYP (W)	MAX (W)
AP3429/A_POWER-CONSUMPTION_MB	2,64	-0,05
LM3671_POWER-CONSUMPTION_MB	1,65	0,94
AP3429/A_POWER-CONSUMPTION_BB	2,45	0,02
LM3671_POWER-CONSUMPTION_BB	1,01	1,0080469

CURRENT CONSUMPTION BUDGET

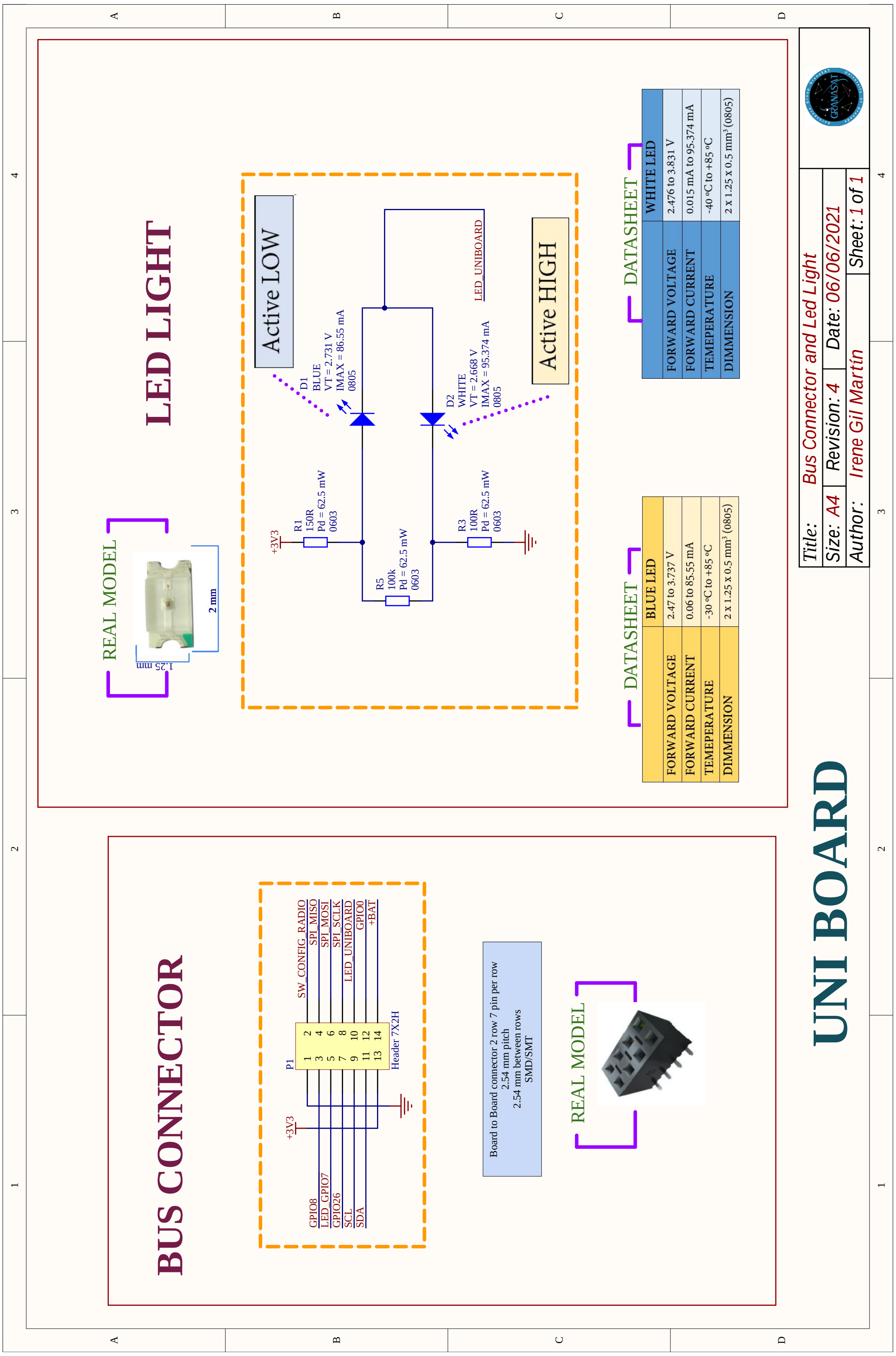
Title: Current Consumption		
Size: A4	Revision: 4	Date: 18/08/2021
Author: Irene Gil Martín		Sheet: 18 of 18

B.2.2 Power Board SCH

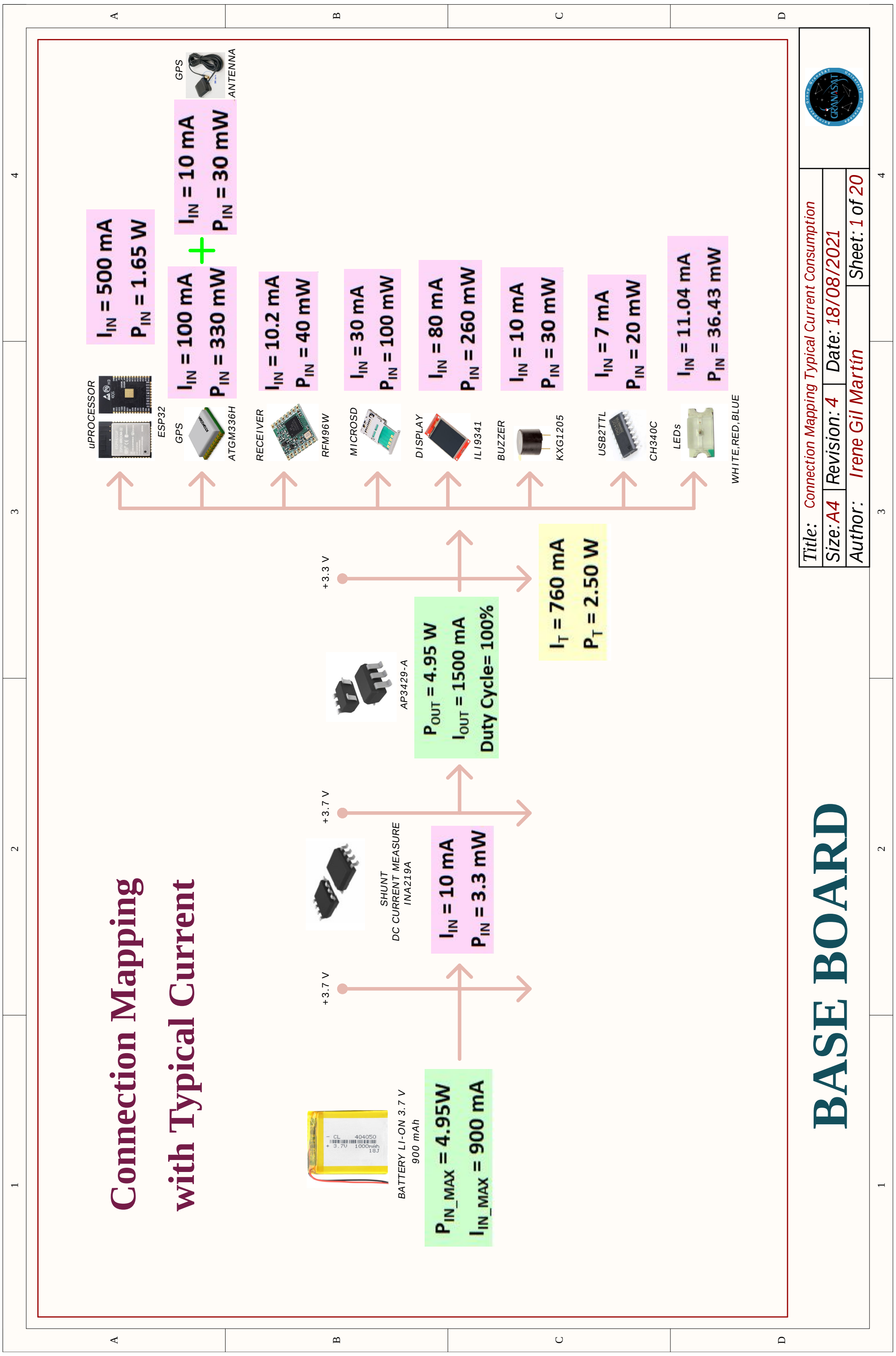


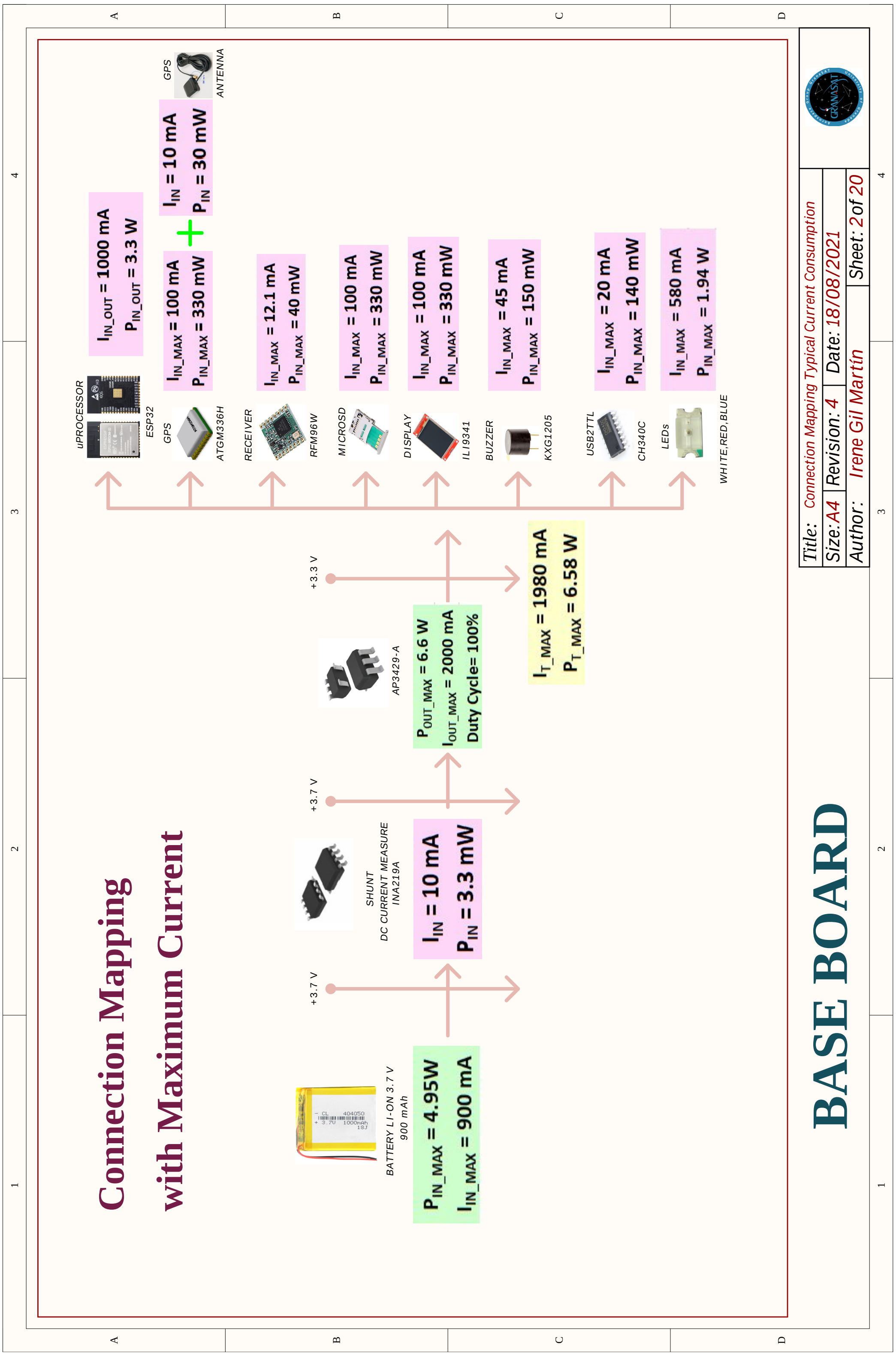

FOOTPRINT ANTENNA GPS
REAL MODEL
X1
Antetenna
Board to Board connector 2 row 7 pin per row
2.54 mm pitch
2.54 mm between rows
SMD/SMT
BUS CONNECTOR
REAL MODEL
P1
Header 7X2H
+3V3
SW_CONFIG_RADIO
SPI_MISO
SPI_MOSI
SPI_SCLK
LED_UNIBOARD
GPIO0
+BAT
GPIO8
LED_GPIO7
GPIO26
SCL
SDA
POWER BOARD
Title: BUS CONNECTOR
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 2 of 2

B.2.3 Uni Board SCH



B.2.4 Base Board SCH




Connection Mapping
with Maximum Current
BATTERY LI-ON 3.7 V
900 mAh
P_IN_MAX = 4.95W
I_IN_MAX = 900 mA
+3.7 V
SHUNT
DC CURRENT MEASURE
INA219A
I_IN = 10 mA
P_IN = 3.3 mW
+3.7 V
AP3429-A
P_OUT_MAX = 6.6 W
I_OUT_MAX = 2000 mA
Duty Cycle= 100%
+3.3 V
I_T_MAX = 1980 mA
P_T_MAX = 6.58 W
uPROCESSOR
ESP32
I_IN_OUT = 1000 mA
P_IN_OUT = 3.3 W
GPS
ATGM336H
I_IN_MAX = 100 mA
P_IN_MAX = 330 mW
GPS
ANTENNA
I_IN = 10 mA
P_IN = 30 mW
RECEIVER
RFM96W
I_IN_MAX = 12.1 mA
P_IN_MAX = 40 mW
MICROSD
I_IN_MAX = 100 mA
P_IN_MAX = 330 mW
DISPLAY
ILI9341
I_IN_MAX = 100 mA
P_IN_MAX = 330 mW
BUZZER
KXG1205
I_IN_MAX = 45 mA
P_IN_MAX = 150 mW
USB2TTL
CH340C
I_IN_MAX = 20 mA
P_IN_MAX = 140 mW
LEDs
WHITE,RED,BLUE
I_IN_MAX = 580 mA
P_IN_MAX = 1.94 W
BASE BOARD
Title: Connection Mapping Typical Current Consumption
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 2 of 20
GRANASAT

uProcessor Pin	Conexion to Module	Information
LED_TX_RX	Blue and White LED White -> '0' Blue -> '1'	A blue led is illuminated when the uProcessor is receiving information through the RX0 pin. A white led is illuminated when the uProcessor is sending information through the TX0 pin.
LED_RADIO	White LED -> '0'	A white led is illuminated when the radio is transmitting
LED_3V3	Blue LED -> '0'	A blue led is illuminated when the uProcessor is connected to the power board, and it is receiving 3.3 V (only if it is charged)
LED_USB_CONNECTED	Red LED -> '0'	A red led is illuminated if the USB is connected.
LED_1PPS	Blue LED -> '0'	A blue led is illuminated every second marked by the 1ppps pin of the GPS module
LED_UNIBOARD	Blue and White LED White -> '0' Blue -> '1'	A blue and white is illuminated in the UniBoard.
SPI_SCLK	MicroSD Radio	Connection through SPI serial clock between the uProcessor and the microSD/Radio.
SPI_MISO	MicroSD Radio	Connection through SPI master in slave out between the uProcessor and the microSD/Radio.
SPI_MOSI	MicroSD Radio	Connection through SPI master out slave in between the uProcessor and the microSD/Radio.
SPI_SS	MicroSD Radio	Connection through SPI slave selection between the uProcessor and the microSD/Radio.
RESET_RADIO	Radio -> '0'	Resetting pin for the Radio Module.
RESET_GPS	GPS -> '0'	Resetting pin for the GPS Module.
SHUTOFF_CONTROL_GPS	GPS	Shutoff control of module, low level effectively

uProcessor Pin	Conexion to Module	Information
SHUTOFF_CONTROL_GPS	GPS	Shutoff control of module, low level effectively
RESET_GPS	GPS -> '0'	Resetting pin for the GPS Module.
SDA	GPS	Connection through I2C bus to send information between the GPS
SCL	GPS	Connection through I2C bus to synchronize the GPS and the uProcessor to send information through the SDA and
SW_TXD	GPS	Serial interface bus to transmit data (UART) from the uProcessor to the GPS or vice versa, from the uProcessor to de CH340C or vice versa
TXD0	CH340C	
SW_RXD	GPS	Serial interface bus to receive data (UART) from the uProcessor to the GPS or vice versa, from the uProcessor to de CH340C or vice versa
RXD0	CH340C	
SW_CONFIG_RADIO	RADIO	Pin connected for software configuration
1PPS	GPS	A 1 pulse per second pulse emitted by the GPS
GPIO0	Programming	Pin used for programming the module
BUZZER	Electromagnetic Buzzer	A buzzer signal activated for emitting a sound
CARD_DETECT	MicroSD	Pin in order to detect when there is a card inserted.
+3V3	All	A 3V3 tension given by the buck converter which is responsible of converting the 3.7 V given by the battery to 3V3 for supplying all the modules
GND	All	Ground connection

USB PIN	Module	Information
D_P	CH340C	Differential signal from the USB positive port
D_N	CH340C	Differential signal from the USB negative port
+5V	Power Board	+5V tension given from the USB to feed the charger circuit
GND	All	Ground Connection

Battery PIN	Module	Information
BAT	INA 219A	3.7 V tension given by the battery

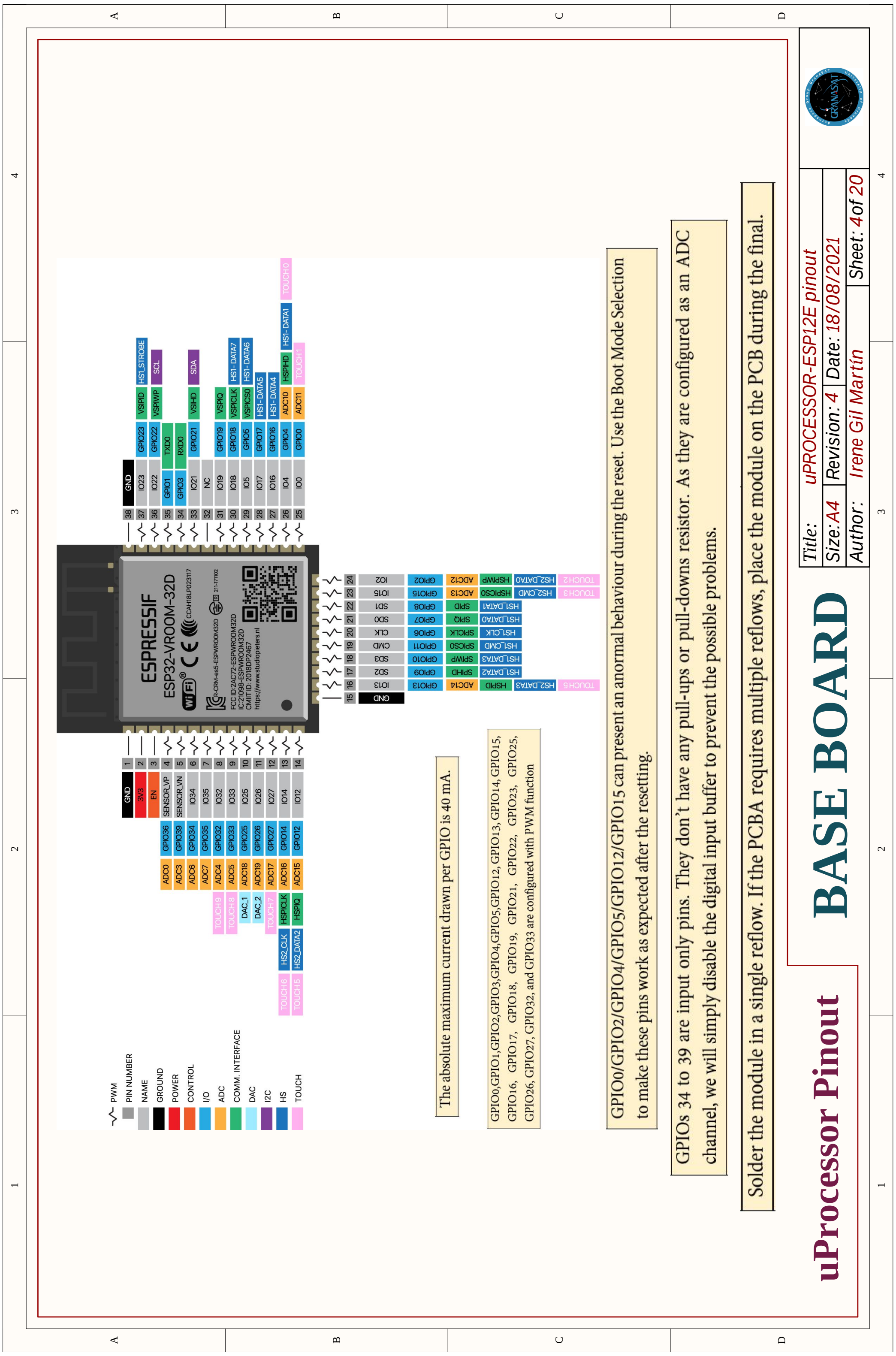
Battery PIN	Module	Information
BAT	INA 219A	3.7 V tension given by the battery

uProcessor Pin	Information
+3V3	A 3V3 tension given by the buck converter which is responsible of converting the 3.7 V given by the battery to 3V3 for supplying all the modules

Connection Mapping

BASE BOARD

Title: Connection Mapping
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martín Sheet: 3 of 20


PWM
PIN NUMBER
NAME
GROUND
POWER
CONTROL
I/O
ADC
COMM. INTERFACE
DAC
I2C
HS
TOUCH
ESPRESSIF
ESP32-VROOM-32D
The absolute maximum current drawn per GPIO is 40 mA.
GPIO0,GPIO1,GPIO2,GPIO3,GPIO4,GPIO5,GPIO12,GPIO13,GPIO14,GPIO15, GPIO16, GPIO17, GPIO18, GPIO19, GPIO21, GPIO22, GPIO23, GPIO25, GPIO26, GPIO27, GPIO32, and GPIO33 are configured with PWM function
GPIO0/GPIO2/GPIO4/GPIO5/GPIO12/GPIO15 can present an anormal behaviour during the reset. Use the Boot Mode Selection to make these pins work as expected after the resetting.
GPIOs 34 to 39 are input only pins. They don't have any pull-ups or pull-downs resistor. As they are configured as an ADC channel, we will simply disable the digital input buffer to prevent the possible problems.
Solder the module in a single reflow. If the PCBA requires multiple reflows, place the module on the PCB during the final.
uProcessor Pinout
BASE BOARD
Title: uPROCESSOR-ESP12E pinout
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 4 of 20

DATASHEET

	ESP-32-WROMM-32D
CPU	Xtensa LX6 32 bits Dual core from 80 MHz to 240 MHz
SRAM	520 kB (8 kB SRAM in RTC)
FLASH	4 MB (larger available)
ROM	448 kB
POWER SUPPLY	3.0 to 3.6 V
CURRENT CONSUMPTION	Minimum current required to be delivered: 500 mA Average Current Consumption:80mA Sleep: 2.5 µA (10 µA RTC + RTC memory) Low Consumption: >150 µA
TEMPERATURE RANGE	-40 to 85 ºC
Wi-Fi	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 150 Mbps
Wi-Fi MODES	Station / SOFTAP / SOFTAP + STATION / P2P
NETWROK PROTOCOL	Ipv4 / Ipv6 / SSL / TCP / UDP / HTTP / FTP / MQTT
BLUETOOTH	V4.2 BR/EDR+BLE
HW ENCRYPTATION	Yes

GPIO	38
DAC	2 DAC Channels 8 bit
ADC	SAR 12 bits 8 Channels
INTERFACES	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / DAC / TOUCH / PWM / LED/ SD card
TOUCH SENSOR	Yes (8 Channels)
TEMPERATURE SENSOR	Yes
HALL EFFECT SENSOR	Yes
DIMMENSIONS	18 x 25.5 x 3.1 mm³ 8 g
PRICE	2.20 € https://www.mouser.es/

The absolute maximum current drawn per GPIO is 40 mA.

GPIO0/GPIO2/GPIO4/GPIO5/GPIO12/GPIO15 can present an anormal behaviour during the reset. Use the Boot Mode Selection to make these pins work as expected after the resetting.

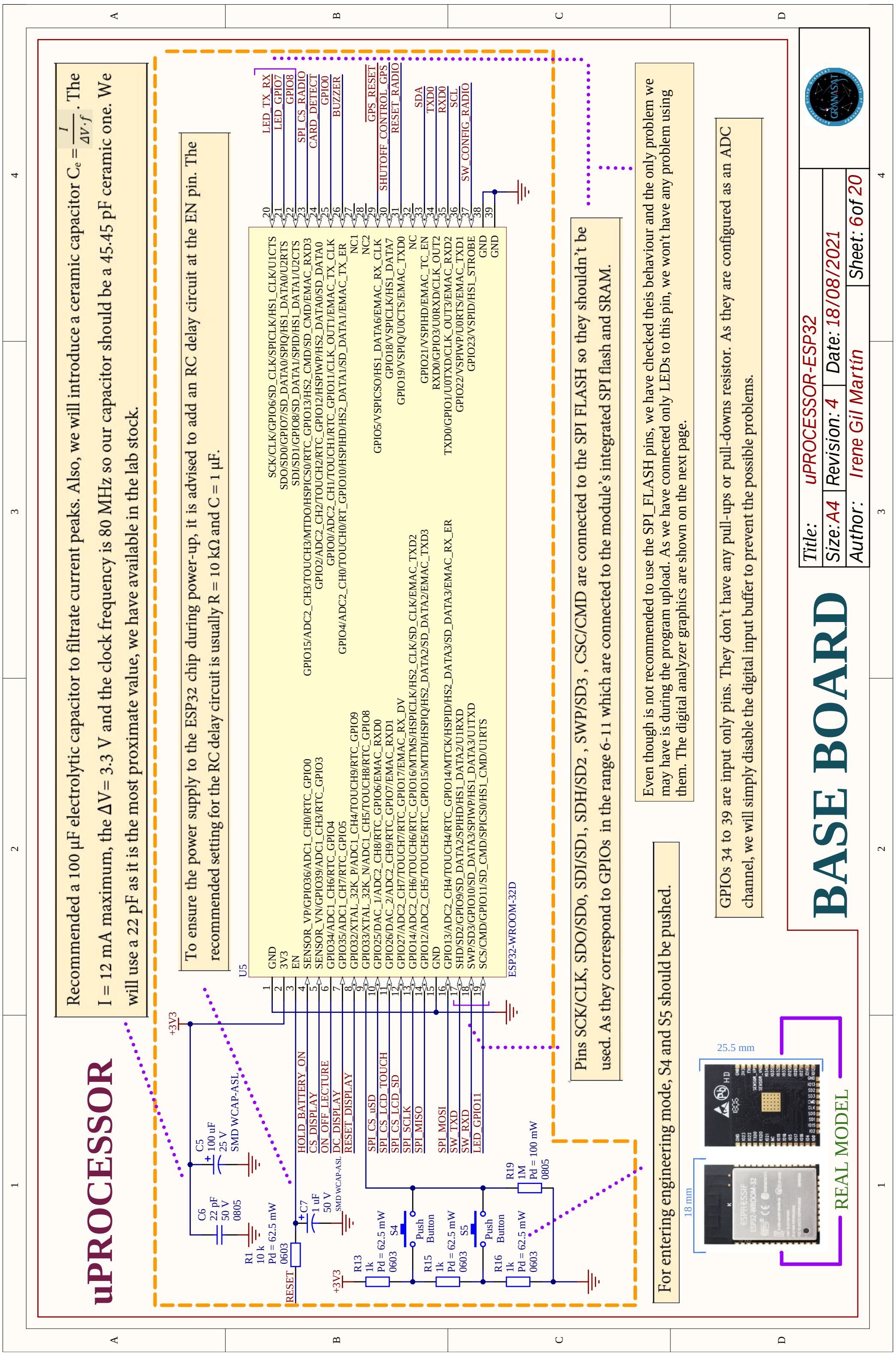
GPIOs 34 to 39 are input only pins. They don't have any pull-ups or pull-downs resistor. As they are configured as an ADC channel, we will simply disable the digital input buffer to prevent the possible problems.

Solder the module in a single reflow. If the PCBA requires multiple reflows, place the module on the PCB during the final.

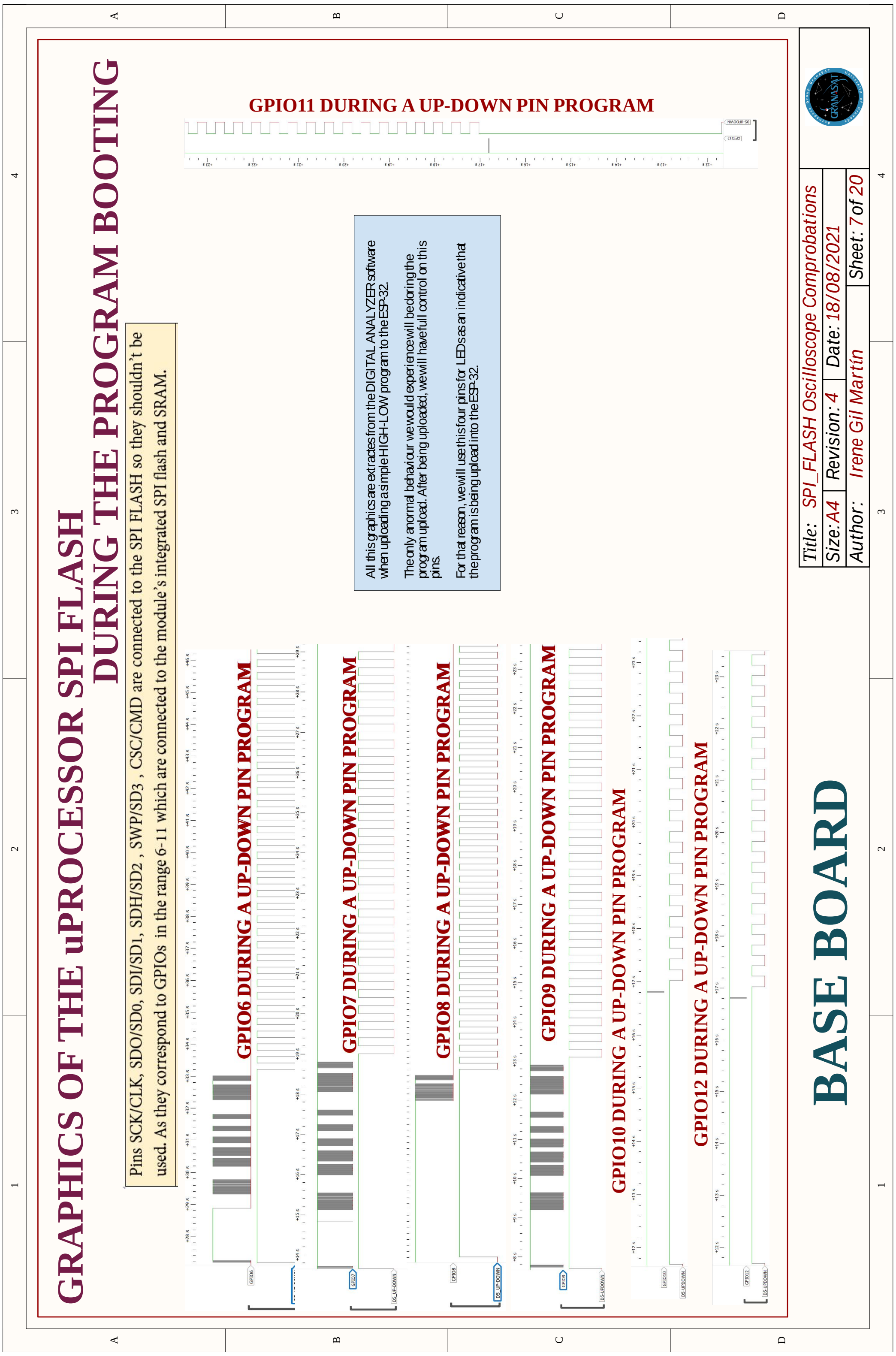
uPROCESSOR

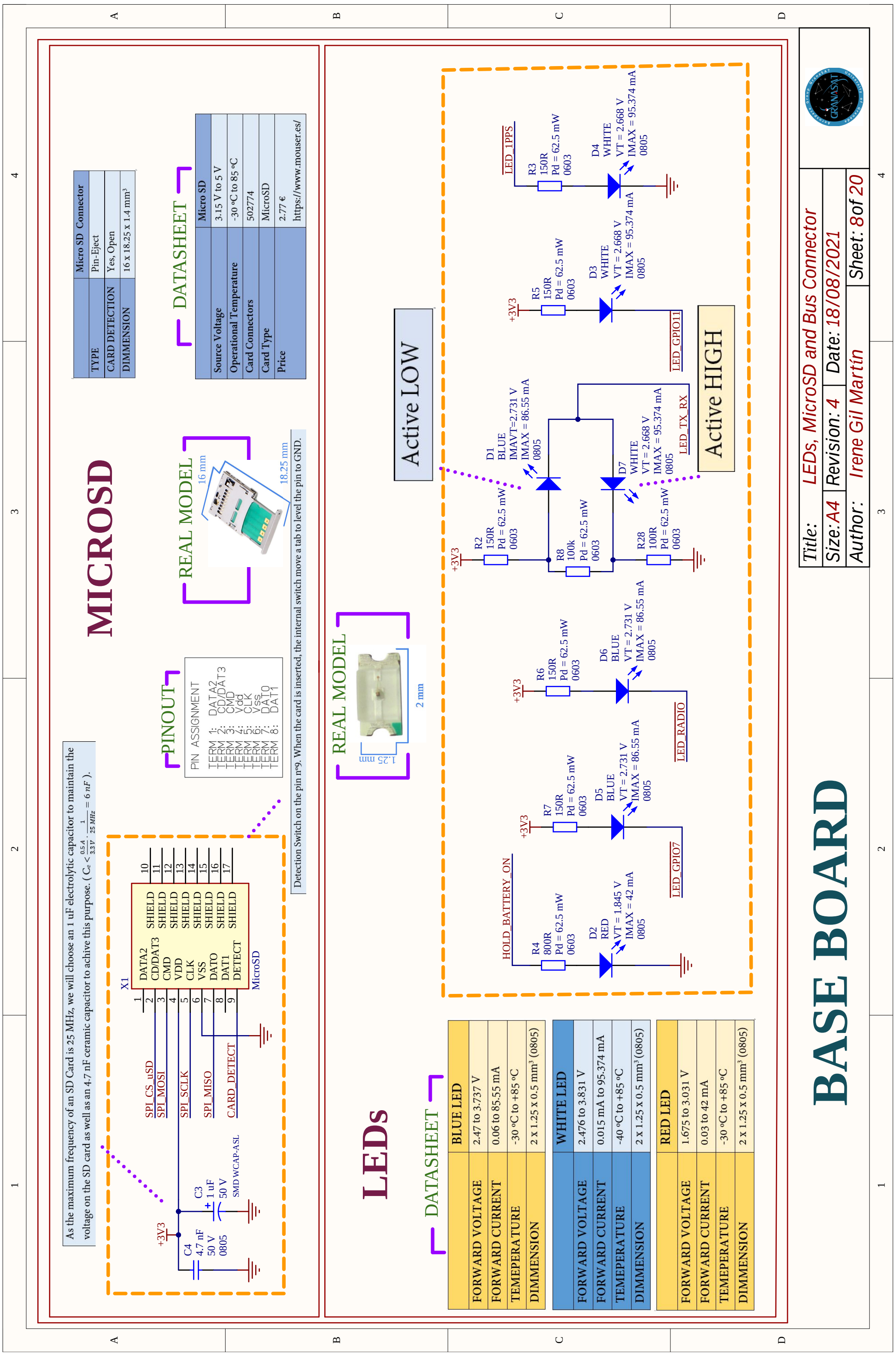
BASE BOARD

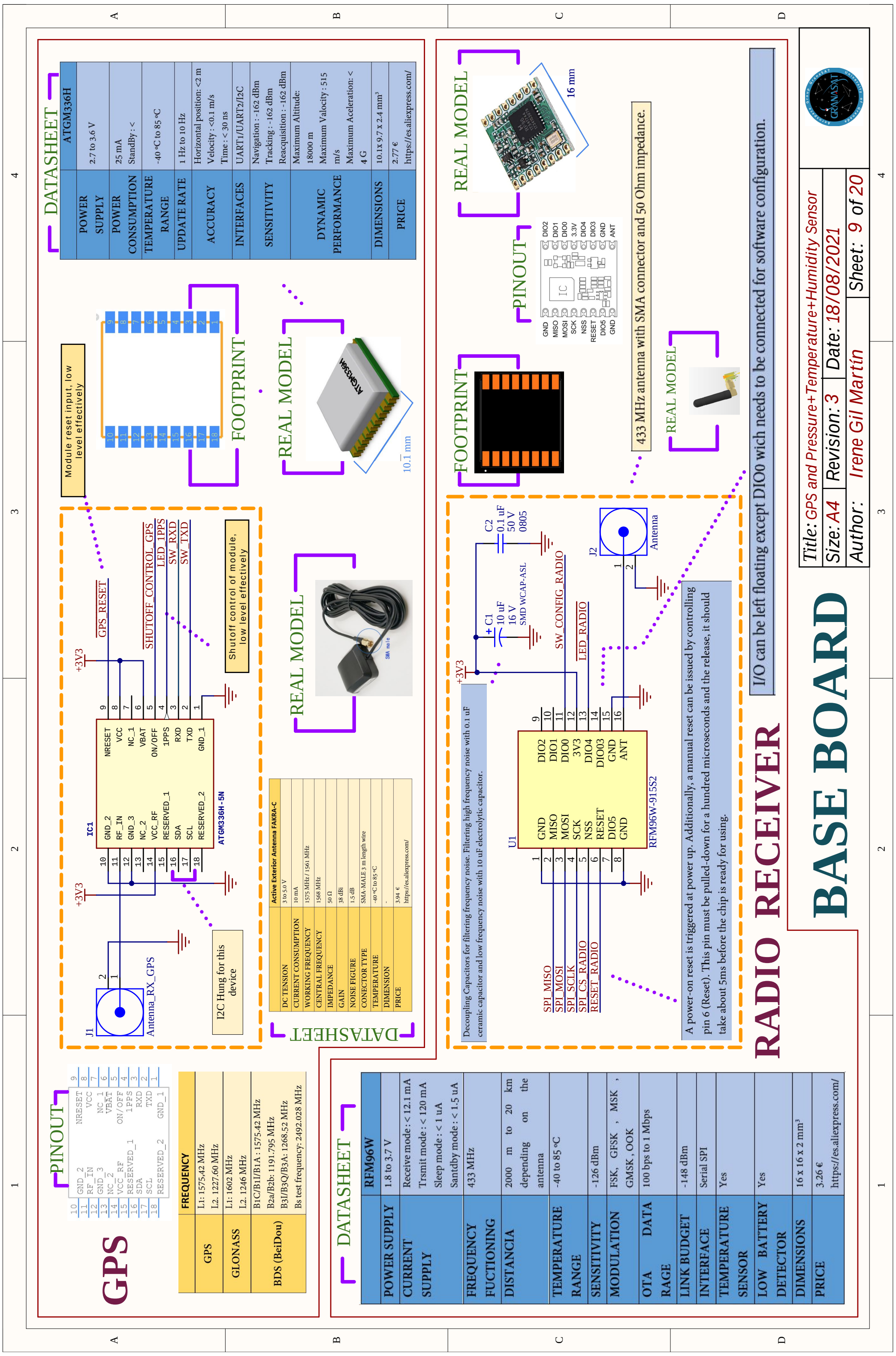
Title: uPROCESSOR-ESP32 information		
Size: A4	Revision: 4	Date: 18/08/2021
Author: Irene Gil Martín		Sheet: 5 of 20

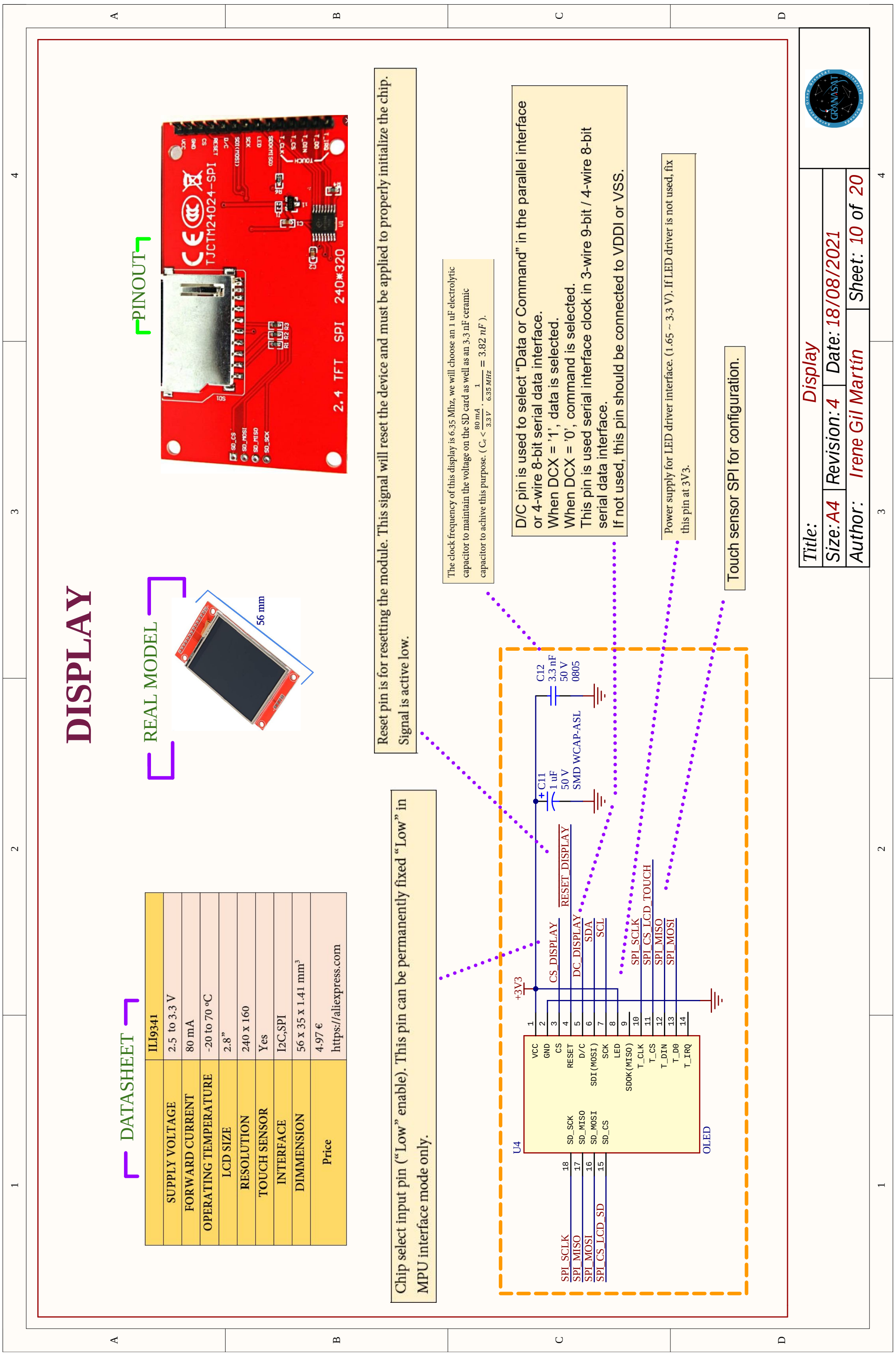


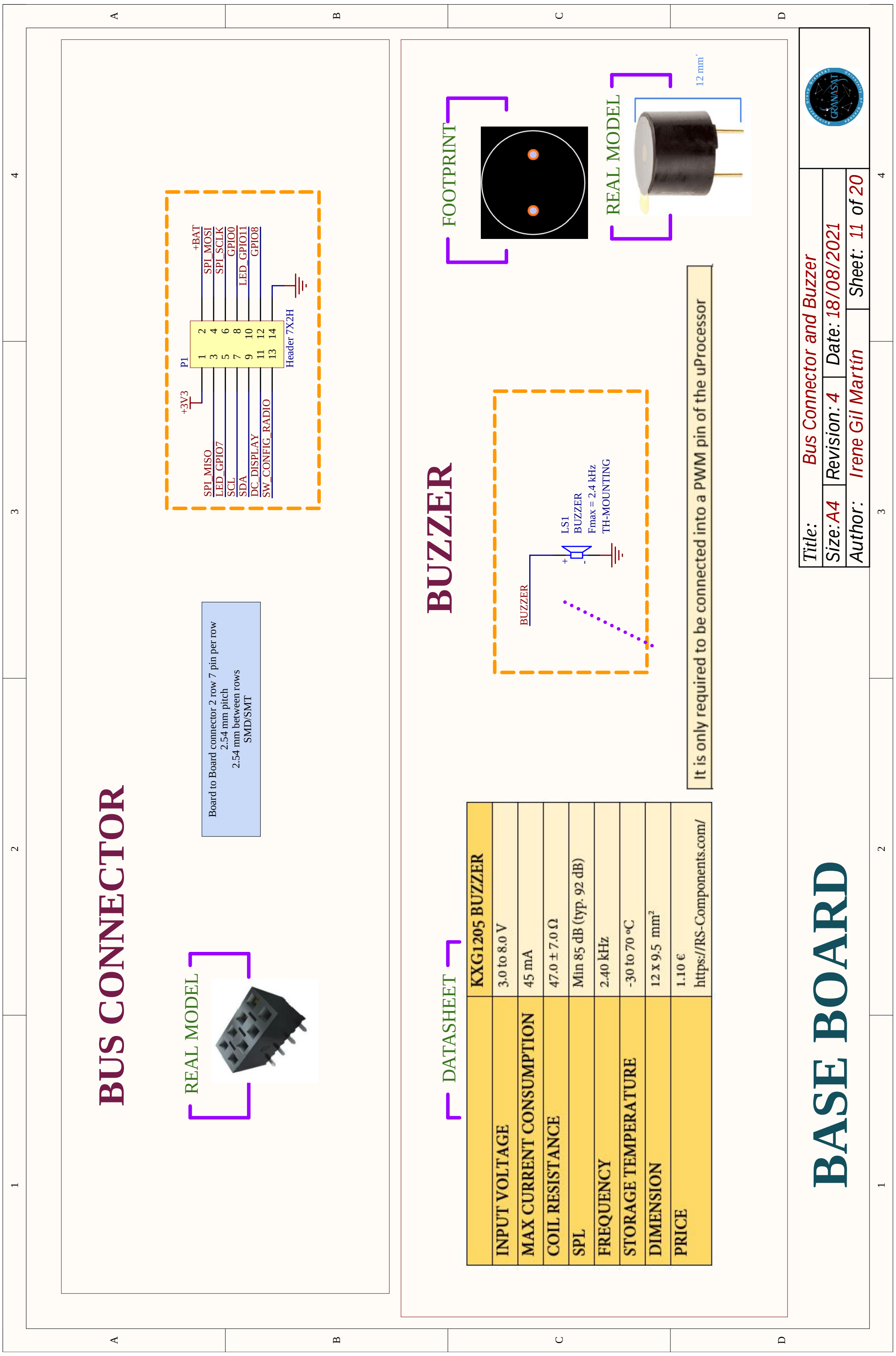
uPROCESSOR
Recommended a 100 µF electrolytic capacitor to filtrate current peaks. Also, we will introduce a ceramic capacitor $C_e = \frac{I}{\Delta V \cdot f}$. The I = 12 mA maximum, the ΔV = 3.3 V and the clock frequency is 80 MHz so our capacitor should be a 45.45 pF ceramic one. We will use a 22 pF as it is the most proximate value, we have available in the lab stock.
To ensure the power supply to the ESP32 chip during power-up, it is advised to add an RC delay circuit at the EN pin. The recommended setting for the RC delay circuit is usually R = 10 kΩ and C = 1 µF.
Pins SCK/CLK, SDO/SD0, SDI/SD1, SDH/SD2 , SWP/SD3 , CSC/CMD are connected to the SPI FLASH so they shouldn't be used. As they correspond to GPIOs in the range 6-11 which are connected to the module's integrated SPI flash and SRAM.
Even though is not recommended to use the SPI_FLASH pins, we have checked theis behaviour and the only problem we may have is during the program upload. As we have connected only LEDs to this pin, we won't have any problem using them. The digital analyzer graphics are shown on the next page.
GPIOs 34 to 39 are input only pins. They don't have any pull-ups or pull-downs resistor. As they are configured as an ADC channel, we will simply disable the digital input buffer to prevent the possible problems.
For entering engineering mode, S4 and S5 should be pushed.
U5
ESP32-WROOM-32D
REAL MODEL
25.5 mm
18 mm
BASE BOARD
Title: uPROCESSOR-ESP32
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 6 of 20


GRAPHICS OF THE uPROCESSOR SPI FLASH DURING THE PROGRAM BOOTING
Pins SCK/CLK, SDO/SD0, SDI/SD1, SDH/SD2 , SWP/SD3 , CSC/CMD are connected to the SPI FLASH so they shouldn't be used. As they correspond to GPIOs in the range 6-11 which are connected to the module's integrated SPI flash and SRAM.
GPIO6 DURING A UP-DOWN PIN PROGRAM
GPIO7 DURING A UP-DOWN PIN PROGRAM
GPIO8 DURING A UP-DOWN PIN PROGRAM
GPIO9 DURING A UP-DOWN PIN PROGRAM
GPIO10 DURING A UP-DOWN PIN PROGRAM
GPIO12 DURING A UP-DOWN PIN PROGRAM
GPIO11 DURING A UP-DOWN PIN PROGRAM
All this graphics are extractes from the DIGITAL ANALYZER software when uploading a simple HIGH-LOW program to the ESP-32.
The only anormal behaviour we would experience will be during the program upload. After being uploaded, we will have full control on this pins.
For that reason, we will use this four pins for LEDs as an indicative that the program is being upload into the ESP-32.
BASE BOARD
Title: SPI_FLASH Oscilloscope Comprobations
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 7 of 20

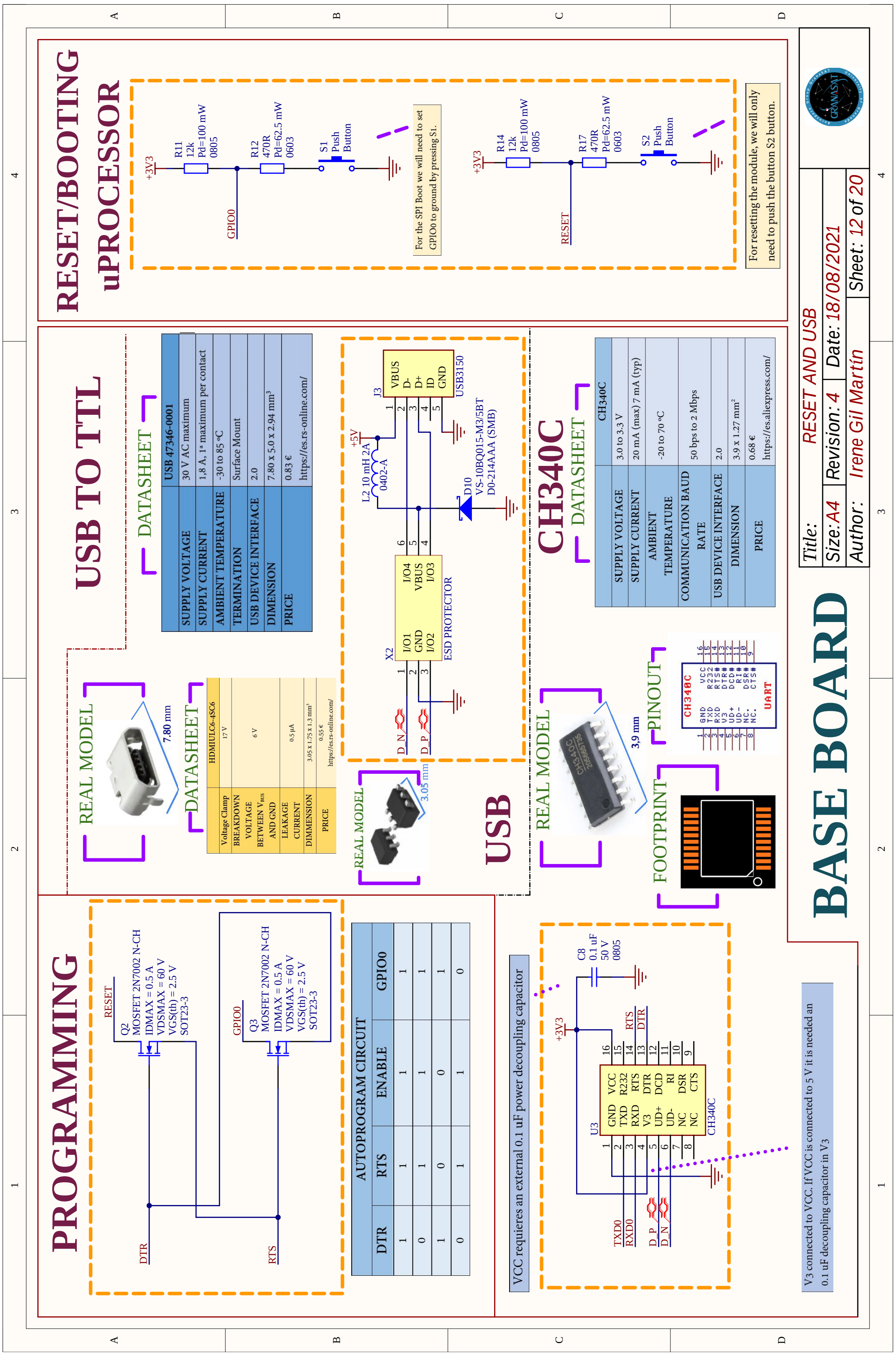

MICROSD
LEDs
Active LOW
Active HIGH
BASE BOARD
Title: LEDs, MicroSD and Bus Connector
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 8of 20

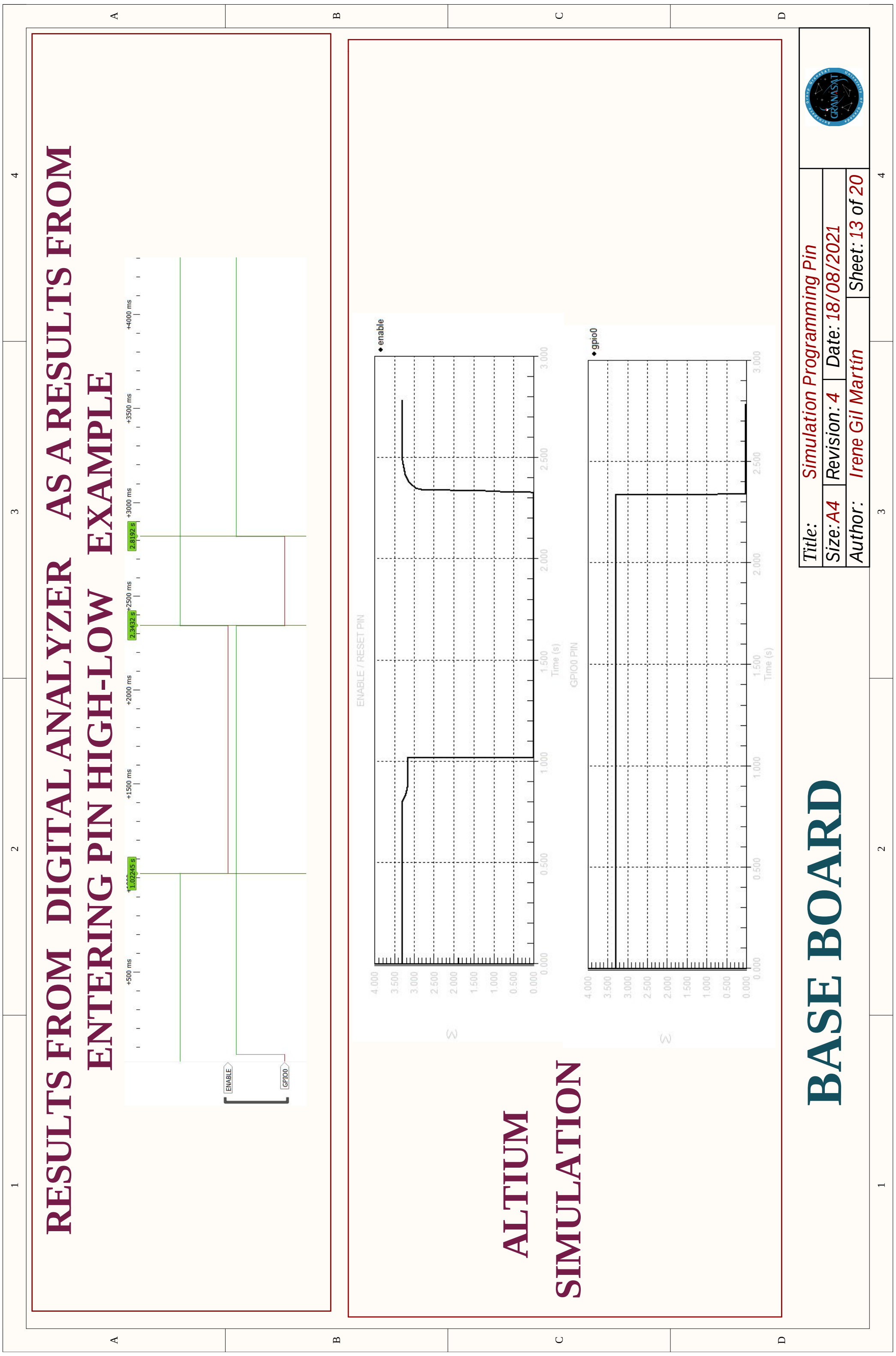

GPS
RADIO RECEIVER
BASE BOARD
DATASHEET
PINOUT
FOOTPRINT
REAL MODEL
Module reset input, low level effectively
Shutoff control of module, low level effectively
I2C Hung for this device
433 MHz antenna with SMA connector and 50 Ohm impedance.
I/O can be left floating except DIO0 wich needs to be connected for software configuration.
A power-on reset is triggered at power up. Additionally, a manual reset can be issued by controlling pin 6 (Reset). This pin must be pulled-down for a hundred microseconds and the release, it should take about 5ms before the chip is ready for using.
Title: GPS and Pressure+Temperature+Humidity Sensor
Size: A4
Revision: 3
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 9 of 20

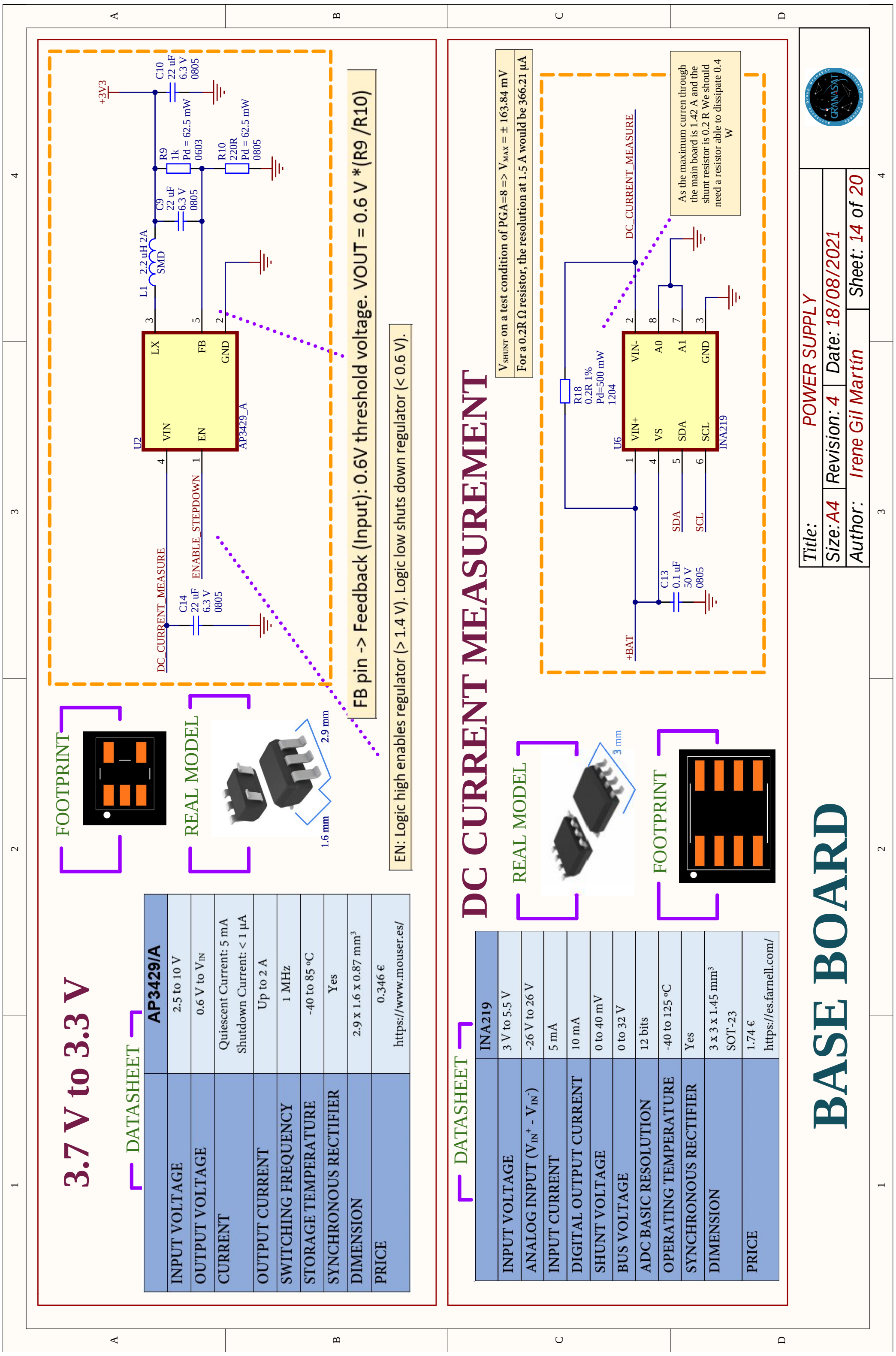

DISPLAY
REAL MODEL
56 mm
PINOUT
DATASHEET
SUPPLY VOLTAGE 2.5 to 3.3 V
FORWARD CURRENT 80 mA
OPERATING TEMPERATURE -20 to 70 ºC
LCD SIZE 2.8"
RESOLUTION 240 x 160
TOUCH SENSOR Yes
INTERFACE I2C,SPI
DIMMENSION 56 x 35 x 1.41 mm³
Price 4.97 € https://aliexpress.com
ILI9341
Reset pin is for resetting the module. This signal will reset the device and must be applied to properly initialize the chip. Signal is active low.
Chip select input pin ("Low" enable). This pin can be permanently fixed "Low" in MPU interface mode only.
D/C pin is used to select "Data or Command" in the parallel interface or 4-wire 8-bit serial data interface.
When DCX = '1', data is selected.
When DCX = '0', command is selected.
This pin is used serial interface clock in 3-wire 9-bit / 4-wire 8-bit serial data interface.
If not used, this pin should be connected to VDDI or VSS.
Power supply for LED driver interface (1.65 ~ 3.3 V). If LED driver is not used, fix this pin at 3V3.
Touch sensor SPI for configuration.
C12 3.3 nF 50 V 0805
C11 1 uF 50 V SMD WCAP-ASL
+3V3
CS_DISPLAY
RESET_DISPLAY
DC_DISPLAY
SDA
SCL
SPI_SCLK
SPI_CS_LCD_TOUCH
SPI_MISO
SPI_MOSI
SPI_CS_LCD_SD
U4
OLED
VCC GND CS RESET D/C SDI(MOSI) SCK LED SDO(MISO) T_CLK T_CS T_DIN T_DO T_IRQ
SD_SCK SD_MISO SD_MOSI SD_CS
Title: Display
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martin Sheet: 10 of 20
GRANASAT

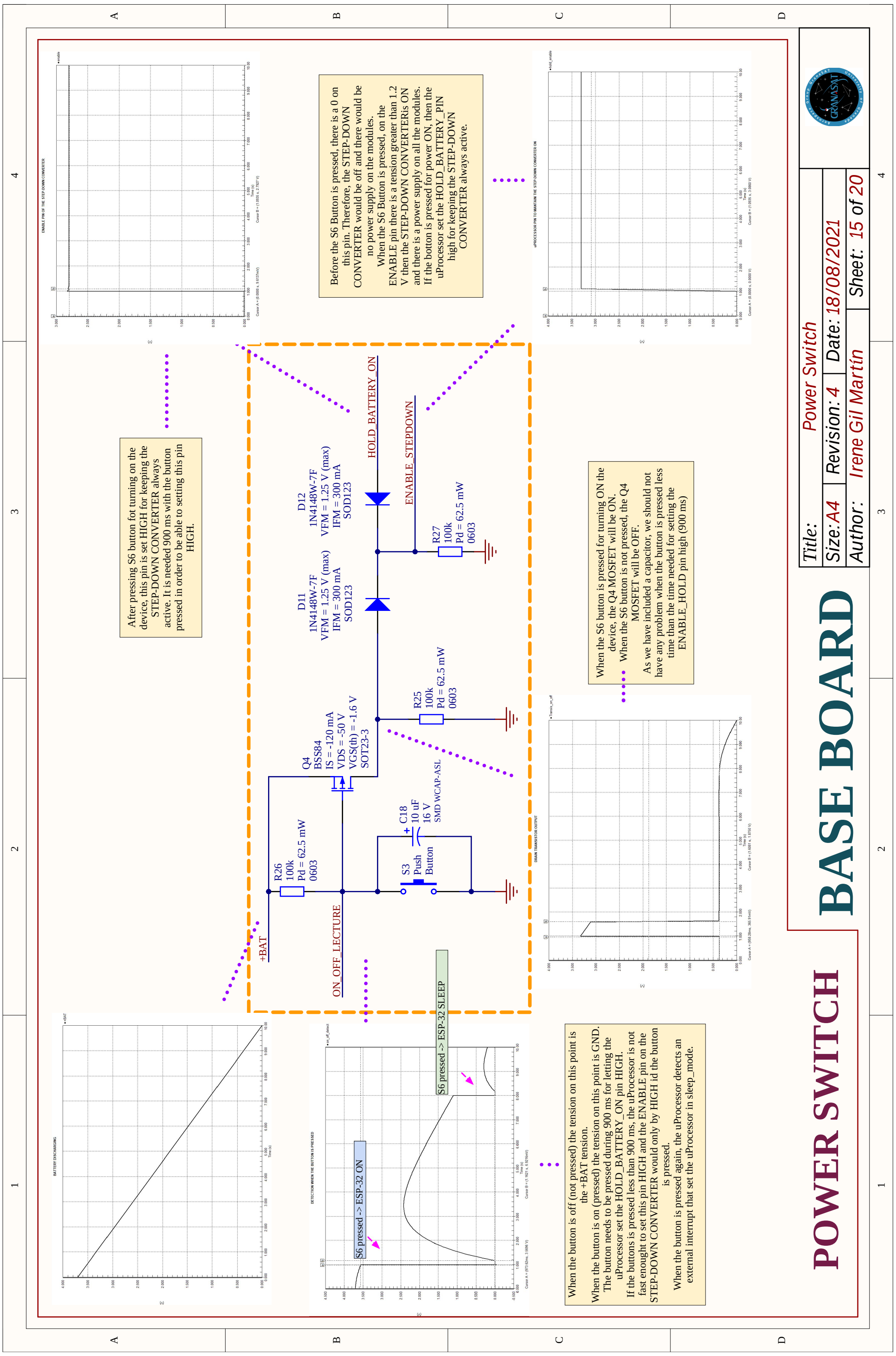


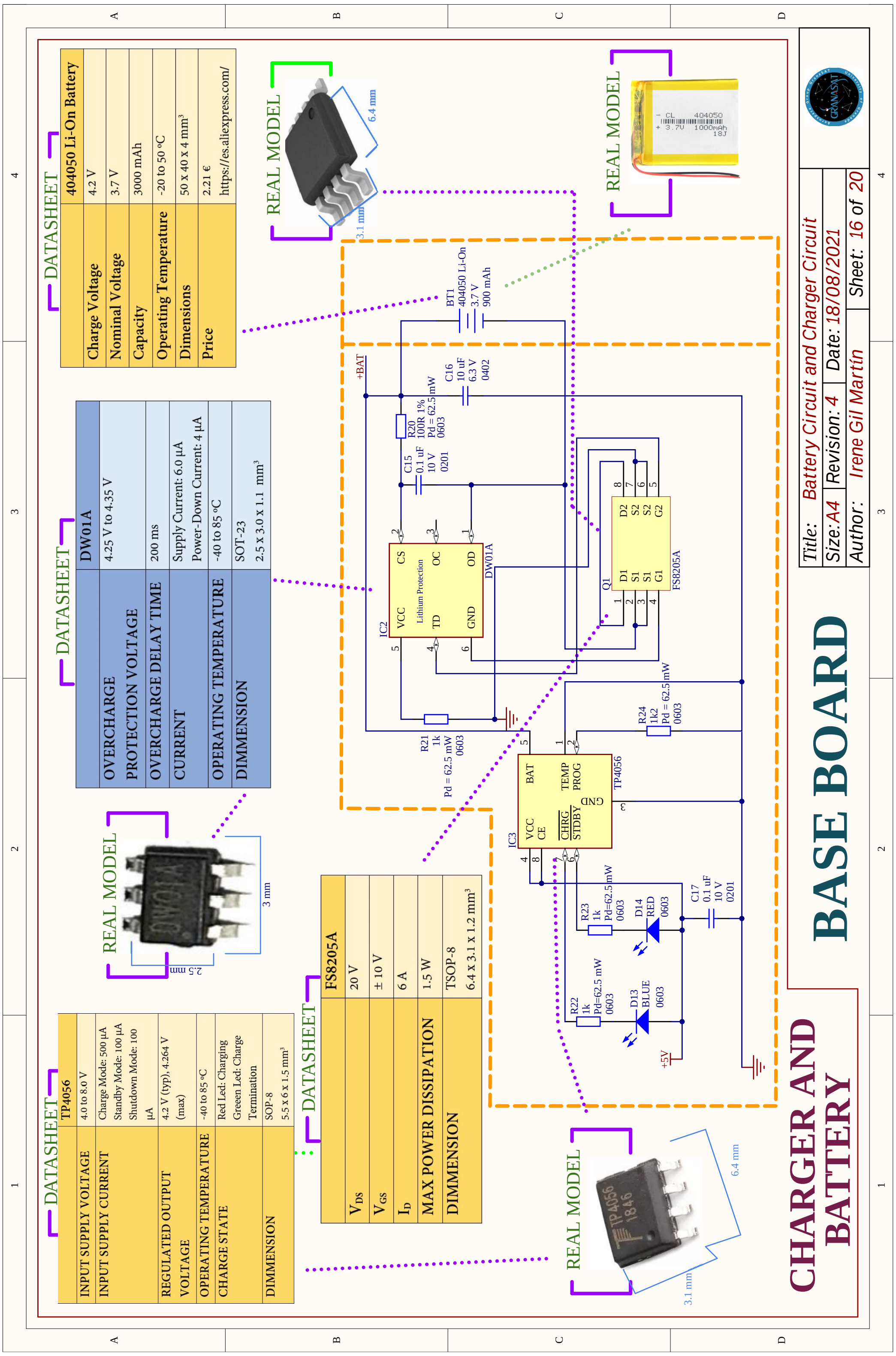
	KXG1205 BUZZER
INPUT VOLTAGE	3.0 to 8.0 V
MAX CURRENT CONSUMPTION	45 mA
COIL RESISTANCE	47.0 ± 7.0 Ω
SPL	Min 85 dB (typ. 92 dB)
FREQUENCY	2.40 kHz
STORAGE TEMPERATURE	-30 to 70 ºC
DIMENSION	12 x 9.5 mm²
PRICE	1.10 €
	https://RS-Components.com/


RESET/BOOTING uPROCESSOR
USB TO TTL
DATASHEET
REAL MODEL
PROGRAMMING
AUTOPROGRAM CIRCUIT
USB
CH340C
FOOTPRINT
PINOUT
BASE BOARD
Title: RESET AND USB
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martin Sheet: 12 of 20
For resetting the module, we will only need to push the button S2 button.
For the SPI Boot we will need to set GPIO0 to ground by pressing S1.
VCC requieres an external 0.1 uF power decoupling capacitor
V3 connected to VCC. If VCC is connected to 5 V it is needed an 0.1 uF decoupling capacitor in V3


RESULTS FROM DIGITAL ANALYZER AS A RESULTS FROM
ENTERING PIN HIGH-LOW EXAMPLE
ENABLE
GPIO0
ALTIUM
SIMULATION
enable
ENABLE / RESET PIN
gpio0
GPIO0 PIN
Time (s)
BASE BOARD
Title: Simulation Programming Pin
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 13 of 20


3.7 V to 3.3 V
DATASHEET
AP3429/A
INPUT VOLTAGE 2.5 to 10 V
OUTPUT VOLTAGE 0.6 V to VIN
CURRENT Quiescent Current: 5 mA Shutdown Current: < 1 µA
OUTPUT CURRENT Up to 2 A
SWITCHING FREQUENCY 1 MHz
STORAGE TEMPERATURE -40 to 85 ºC
SYNCHRONOUS RECTIFIER Yes
DIMENSION 2.9 x 1.6 x 0.87 mm3
PRICE 0.346 € https://www.mouser.es/
FOOTPRINT
REAL MODEL
2.9 mm
1.6 mm
FB pin -> Feedback (Input): 0.6V threshold voltage. VOUT = 0.6 V *(R9 /R10)
EN: Logic high enables regulator (> 1.4 V). Logic low shuts down regulator (< 0.6 V).
U2
AP3429_A
VIN
EN
LX
FB
GND
DC_CURRENT_MEASURE
ENABLE_STEPDOWN
C14 22 uF 6.3 V 0805
L1 2.2 uH 2A SMD
C9 22 uF 6.3 V 0805
R9 1k Pd = 62.5 mW 0603
R10 220R Pd = 62.5 mW 0805
C10 22 uF 6.3 V 0805
+3V3
DC CURRENT MEASUREMENT
DATASHEET
INA219
INPUT VOLTAGE 3 V to 5.5 V
ANALOG INPUT (VIN+ - VIN-) -26 V to 26 V
INPUT CURRENT 5 mA
DIGITAL OUTPUT CURRENT 10 mA
SHUNT VOLTAGE 0 to 40 mV
BUS VOLTAGE 0 to 32 V
ADC BASIC RESOLUTION 12 bits
OPERATING TEMPERATURE -40 to 125 ºC
SYNCHRONOUS RECTIFIER Yes
DIMENSION 3 x 3 x 1.45 mm3 SOT-23
PRICE 1.74 € https://es.farnell.com/
REAL MODEL
3 mm
FOOTPRINT
VSHUNT on a test condition of PGA=8 => VMAX = ± 163.84 mV
For a 0.2R Ω resistor, the resolution at 1.5 A would be 366.21 µA
As the maximum curren through the main board is 1.42 A and the shunt resistor is 0.2 R We should need a resistor able to dissipate 0.4 W
U6
INA219
VIN+
VIN-
VS
SDA
SCL
A0
A1
GND
R18 0.2R 1% Pd=500 mW 1204
C13 0.1 uF 50 V 0805
+BAT
DC_CURRENT_MEASURE
BASE BOARD
Title: POWER SUPPLY
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 14 of 20


POWER SWITCH
BASE BOARD
Title: Power Switch
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martin
Sheet: 15 of 20
GRANASAT
Before the S6 Button is pressed, there is a 0 on this pin. Therefore, the STEP-DOWN CONVERTER would be off and there would be no power supply on the modules.
When the S6 Button is pressed, on the ENABLE pin there is a tension greater than 1.2 V then the STEP-DOWN CONVERTERis ON and there is a power supply on all the modules.
If the botton is pressed for power ON, then the uProcessor set the HOLD_BATTERY_PIN high for keeping the STEP-DOWN CONVERTER always active.
After pressing S6 button fot turning on the device, this pin is set HIGH for keeping the STEP-DOWN CONVERTER always active. It is needed 900 ms with the button pressed in order to be able to setting this pin HIGH.
When the S6 button is pressed for turning ON the device, the Q4 MOSFET will be ON.
When the S6 button is not pressed, the Q4 MOSFET will be OFF.
As we have included a capacitor, we should not have any problem when the button is pressed less time than the time needed for setting the ENABLE_HOLD pin high (900 ms)
When the button is off (not pressed) the tension on this point is the +BAT tension.
When the button is on (pressed) the tension on this point is GND.
The button needs to be pressed during 900 ms for letting the uProcessor set the HOLD_BATTERY_ON pin HIGH.
If the buttons is pressed less than 900 ms, the uProcessor is not fast enought to set this pin HIGH and the ENABLE pin on the STEP-DOWN CONVERTER would only by HIGH id the button is pressed.
When the button is pressed again, the uProcessor detects an external interrupt that set the uProcessor in sleep_mode.
HOLD_BATTERY_ON
ENABLE_STEPDOWN
+BAT
ON_OFF_LECTURE
D12
1N4148W-7F
VFM = 1.25 V (max)
IFM = 300 mA
SOD123
D11
1N4148W-7F
VFM = 1.25 V (max)
IFM = 300 mA
SOD123
Q4
BSS84
IS = -120 mA
VDS = -50 V
VGS(th) = -1.6 V
SOT23-3
R27
100k
Pd = 62.5 mW
0603
R25
100k
Pd = 62.5 mW
0603
R26
100k
Pd = 62.5 mW
0603
C18
10 uF
16 V
SMD WCAP-ASL
S3
Push
Button
S6 pressed -> ESP-32 SLEEP
S6 pressed -> ESP-32 ON
A
B
C
D
1
2
3
4


DATASHEET
404050 Li-On Battery
Charge Voltage 4.2 V
Nominal Voltage 3.7 V
Capacity 3000 mAh
Operating Temperature -20 to 50 ºC
Dimensions 50 x 40 x 4 mm³
Price 2.21 €
https://es.aliexpress.com/
DATASHEET
DW01A
OVERCHARGE PROTECTION VOLTAGE 4.25 V to 4.35 V
OVERCHARGE DELAY TIME 200 ms
CURRENT Supply Current: 6.0 µA Power-Down Current: 4 µA
OPERATING TEMPERATURE -40 to 85 ºC
DIMMENSION SOT-23 2.5 x 3.0 x 1.1 mm³
DATASHEET
TP4056
INPUT SUPPLY VOLTAGE 4.0 to 8.0 V
INPUT SUPPLY CURRENT Charge Mode: 500 µA Standby Mode: 100 µA Shutdown Mode: 100 µA
REGULATED OUTPUT VOLTAGE 4.2 V (typ), 4.264 V (max)
OPERATING TEMPERATURE -40 to 85 ºC
CHARGE STATE Red Led: Charging Green Led: Charge Termination
DIMMENSION SOP-8 5.5 x 6 x 1.5 mm³
DATASHEET
FS8205A
V_DS 20 V
V_GS ± 10 V
I_D 6 A
MAX POWER DISSIPATION 1.5 W
DIMMENSION TSOP-8 6.4 x 3.1 x 1.2 mm³
REAL MODEL
REAL MODEL
REAL MODEL
REAL MODEL
BT1 404050 Li-On 3.7 V 900 mAh
+BAT
IC2 DW01A Lithium Protection
Q1 FS8205A
IC3 TP4056
R20 100R 1% Pd = 62.5 mW 0603
C15 0.1 uF 10 V 0201
C16 10 uF 6.3 V 0402
R21 1k Pd = 62.5 mW 0603
R24 1k2 Pd = 62.5 mW 0603
R23 1k Pd=62.5 mW 0603
R22 1k Pd=62.5 mW 0603
D14 RED 0603
D13 BLUE 0603
C17 0.1 uF 10 V 0201
+5V
CHARGER AND BATTERY
BASE BOARD
Title: Battery Circuit and Charger Circuit
Size: A4
Revision: 4
Date: 18/08/2021
Author: Irene Gil Martín
Sheet: 16 of 20

uPROCESSOR

	SAMD21G18A-AUT	ESP-32	ESP-12
CPU	ARM CORTEX -M0+CPU running at up to 48 MHz: - Single-cycle hardware multiplier - Micro Trace Buffer (MTB)	Xtensa LX6 32 bits Single/Dual core Up to 600 MIPS	Tensilica L X106 32 bits Single core
SRAM	32 kB	520 kB (16 kB SRAM in RTC)	< 50 kB
FLASH	256	4 MB (larger available)	4 MB
ROM	No	448 kB	No Porgrammable ROM
POWER SUPPLY	1.62 to 3.63 V	2.3 to 3.6 V	3.0 to 3.6 v
CURRENT CONSUMPTION	85 to 1.2 mA	Max: 225 mA Typ:80mA Sleep: 2.5 µA (10 µA RTC + memoria RTC) Low Consumption: >150 µA	Max: 225 mA Typ:80mA Sleep: 20 µA (RTC + memoria RTC)
TEMPERATURE RANGE	-40 to 125 ºC	-40 to 125 ºC	-40 to 125 ºC
Wi-Fi	No	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 150 Mbps	802.11 b/g/n (up to +20 dBm) WEP,WPA 2.4 GHz Up to 72.2Mbps
Wi-Fi MODES	-	Station / SOFTAP / SOFTAP + STATION / P2P	Station / SOFTAP / SOFTAP + STATION / P2P
NETWROK PROTOCOL	-	Ipv4 / Ipv6 / SSL / TCP / UDP / HTTP / FTP / MQTT	Ipv4 / TCP / UDP / HTTP / MQTT
BLUETOOTH	No	V4.2 BR/EDR+BLE	No
ETHERNET MAC INTERFACE	-	10/100 Mbps	No

	SAMD21G18A-AUT	ESP-32	ESP-12
HW ENCRYPTATION	No	Yes	No (TLS 1.2 in SW)
GPIO	Up to 52	36	17
DAC	10 bit	2 DAC channels 8 bit	NO
ADC	12 bit Up to 20 Channels	SAR 12 bits 8 Channels	SAR 10 bits 1 Channel
INTERFACES	DMAC / TC / TCC / RTC / WDT / CRC / SERCOM / USART / 12C / SPI / ADC / DAC / AC / PTC	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / DAC / TOUCH / PWM / LED	UART / SDIO / SPI / I2C / I2S / IR REMOTE CONTROL GPIO / ADC / PWM / LED
TOUCH SENSOR	Yes	Yes (8 Channels)	No
TEMPERATURE SENSOR	No	Yes	No
HALL EFFECT SENSOR	No	Yes	No
DIMMENSIONS	7 x 7 x 1.0 mm³	57 x 28 x 1 mm³ 8 g	16 x 24 x 3.4 mm³ 1.72 g
PRICE	2.73 €	2.20 € https://www.mouser.es/	3 – 6€
PREFERENCE ORDER	3	1	2
ELECTION	✗	✓	✗

DEVICES TRADE-OFF

Title: uProcessor's and Display Comparative
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martín Sheet: 17 of 20

GPS

	Ublox-MAX8C	ATGM336H	CAM-M8C	BG01-H111S100
POWER SUPPLY	1.65 – 3.6 V	2.7 to 3.6 V	1.65 TO 3.6 V	Typ. 3.3 V
POWER CONSUMPTION	Low power consumption	25 mA StandBy : <	71 mA	Acquisition Current: 30 mA Tracking Current: 20 mA
TEMPERATURE RANGE	-40 ºC to 85 ºC	-40 ºC to 85 ºC	-40 ºC to 85 ºC	-40 ºC to 85 ºC
UPDATE RATE	0.25 Hz to 10 MHz	1 Hz to 10 Hz	10 Hz	
ACCURACY	Horizontal position <4 m Velocity : <0.05 m/s Time:60 ns	Horizontal position: <2 m Velocity : <0.1 m/s Time : < 30 ns	Horizontal position: Velocity : <0.05 m/s Acceleration:	Horizontal position: 2.5 m Altitude position: 3.5 m Velocity : 0.1 m/s Time pulse signal: 30 ns
INTERFACES	UART , DDC (I2C fast mode compatible)	UART1/UART2/I2C	UART / DDC / I2C	UART
SENSITIVITY	Navigation : -166 dBm Tracking : -166 dBm Reacquisition : -160 dBm	Navigation : -162 dBm Tracking : -162 dBm Reacquisition : -162 dBm	Navigation : -164 dBm Tracking : -164 dBm Reacquisition : -160 dBm	Cold Start : -148 dBm Warm Start: -162 dBm Tracking : -166 dBm Reacquisition : -164 dBm
DYNAMIC PERFORMANCE	Maximum Altitude: 50000 m Maximum Valocity : 500 m/s Maximum Aceleration: < 4 G	Maximum Altitude: 18000 m Maximum Valocity : 515 m/s Maximum Aceleration: < 4 G	Maximum Altitude: 50000 m Maximum Valocity : 500 m/s Maximum Aceleration: < 4 G	Cold Start: 27.5 s Warm Start: < 1 s Re-Acquisition: < 1 s A-GNSS: < 1 s
DIMENSIONS	9.7 x 10.1 mm²	10.1x 9.7 x 2.4 mm³ 0.6 g	9.60 x 14 x 2 mm³	16.2 x 12.2 mm²
PRICE	5.62 € https://es.aliexpress.com/	2.77 € https://es.aliexpress.com/		6.75 € www.mecter.com
PREFERENCE ORDER	4	1	3	2
ELECTION	✗	✓	✗	✗

RADIO RECEIVER

	RFM69W	SX1278	RFM96W	HM-TRP
POWER SUPPLY	2.4 to 3.6 Vdc	1.8 to 3.7 V	1.8 to 3.7 V	2.4 to 3.6 V
CURRENT SUPPLY	Receive mode : 16 mA Trsmit mode : 16 – 45 mA Sleep mode : < 1 uA Santdby mode : <1.5 mA	Receive mode : <12 mA Trsmit mode : < 20 mA Sleep mode : < 1 uA Santdby mode : <1.8 mA	Receive mode : < 12.1 mA Trsmit mode : < 120 mA Sleep mode : < 1 uA Santdby mode : < 1.5 uA	Receive mode : < 30 mA Trsmit mode : < 120 mA Sleep mode : 2 uA
FREQUENCY FUCTIONING	433 MHz	433 MHz	433 MHz	414 to 454 MHz ISM band
DISTANCIA	500 m	5000 m	2000 m to 20 km depending on the antenna	Over 1 km in open air
TEMPERATURE RANGE	- 20 to 70 ºC	-40 to 85 ºC	-40 to 85 ºC	-40 to 85 ºC
SENSITIVITY	-120 dBm	Down to -148 dBm	-126 dBm	-117 dBm
MODULATION	FSK , GFSK , MSK , GMSK or OOK	FSK, GFSK, MSK, GMSK, OOK	FSK, GFSK , MSK , GMSK , OOK	FSK 2 way half-duplex communication, strong anti-interfere
OTA DATA RAGE	1.2 to 300 kbps	1 to 300 kbps	100 bps to 1 Mbps	1.2 kbps to 115.2 kbps
LINK BUDGET	115 dB	168 dB	-148 dBm	
INTERFACE	Serial SPI	Serial SPI	Serial SPI	Standard TTL UART, extendable to RS232 or other interface
TEMPERATURE SENSOR	Yes	Yes	Yes	No
LOW BATTERY DETECTOR	No	Yes	Yes	Yes
DIMENSIONS	19.7 x 16 x 1.9 mm³	17.2 X 17 mm²	16 x 16 x 2 mm³	16 x 20 x 2 mm³
PRICE	1.52 € https://es.aliexpress.com/	3.13 € https://es.aliexpress.com/	3.26 € https://es.aliexpress.com/	7.25 € www.mecter.com
PREFERENCE ORDER	4	3	1	2
ELECTION	✗	✗	✓	✗

DEVICES TRADE-OFF

Title: GPS and Radio Comparative
Size: A4 Revision: 4 Date: 18/08/2021
Author: Irene Gil Martín Sheet: 18 of 20

DEVICES TRADE-OFF

DISPLAY

	LCD + SD 8TFT 128x160	SSD1331 OLED	SSD 1306 OLED	ILI9341	TFT LCD MODULE-HT0280CI01BR1
SUPPLY VOLTAGE	-0.3 to 4.6 V	2.4 to 3.5 V	1.65 to 3.3 V	2.5 to 3.3 V	2.5 to 3.3 V
CURRENT		Sleep Mode: 10 uA Max Supply Current: 500 uA	Sleep Mode: 10 uA Max Supply Current : 150 uA	80 mA	80 mA
TEMPERATURE RANGE	-30 to 85 ºC	-40 to 85 ºC	-40 to 85 ºC	-20 to 70 ºC	-20 to 70 ºC
RESOLUTION	128 x 160 1.8”	96 x 64 0.95”	128 x 64 0.96”	240 x 160 2.8”	240 x 320 2.8”
COLORS	65 thousand RGB format	65 thousand	Monochrome	RGB Format	RGB Format
TOUCH PANNEL	No	No	No	Yes	Yes
DIMMENSION	58 x 34 mm^2	30.7 x 27.3 x 11.3 mm^3	26.7 x 19.26 x 1.85 mm^3	56 x 35 x 1.41 mm^3	50 x 69.2 x 3.45 mm^3
PRICE	21 $ www.hotmcu.com	5.20 € https://es.aliexpress.com/	7.99 € https://es.aliexpress.com/	4.97 € https://aliexpress.com	7.25 € www.mecter.com
PREFERENCE ORDER	3	4	5	1	2
ELECTION	✗	✗	✗	✓	✗

Title: Display Comparative		
Size: A4	Revision: 4	Date: 18/08/2021
Author: Irene Gil Martín		Sheet: 19 of 20

MAXIMUM CONSUMPTION BUDGET					
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT_MAX} (A)	I_{OUT} (A)
AP3428/A (3.7 V -> 3.3 V)	1	3,6	6,6	2	2
LM3671 (3.7 V -> 3.3 V)	2	3,6	3,795	1,15	1,02

TYPICAL CONSUMPTION BUDGET					
Component	Number/PCB	V_{IN} (V)	P_{OUT} (W)	I_{OUT_MAX} (A)	I_{OUT} (A)
AP3429/A (3.7 V -> 3.3 V)	1	3,6	4,95	2	1,5
LM3671 (3.7 V -> 3.3 V)	2	3,6	1,98	0,8	0,6

	TYP	MAX
V_{SOURCE} (V)	3,7	3,7
CAPACITY (mAh)	900	900
SOURCE SPECS	3.7 V / 3.3 V	3.7 V / 3.3 V
TOTAL POWER MAIN BOARD(W)	2,31	6,65
TOTAL POWER BASE BOARD(W)	2,50	6,58
CURRENT NEEDED MAIN BOARD (A)	0,70	1,98
CURRENT NEEDED BASE BOARD (A)	0,76	1,98

Component	Module	Number/PCB	V_{IN} (V)	I_{IN_TYP} (mA)	I_{IN_MAX} (mA)	P_{IN_TYP} (W)	P_{IN_MAX} (W)
uProcessor	ESP-32-WROOM32E	1	3,3	500	1000	1,65	3,30
microSD	SanDisk standard uSD	1	3,3	30	100	0,10	0,33
GPS	ATGM-336H	1	3,3	100	100	0,33	0,33
	Ceramic Antenna	1	3,3	10	10	0,03	0,03
	Exterior Antenna	1	3,3	10	10	0,03	0,03
Radio Transmitter	RFM96W	1	3,3	20	120	0,07	0,40
Radio Receiver	RFM96W	1	3,3	10,8	12,1	0,04	0,04
Pressure				2,80E-03	4,20E-03	9,24E-06	1,39E-05
Humidity Sensor	BME280	1	3,3	1,80E-03	2,80E-03	5,94E-06	9,24E-06
Temperature				1,00E-03	2,00E-03	3,30E-06	3,60E-09
Buzzer	KXG1205	1	3,3	10	45	0,03	0,15
Display	ILI9341	1	3,3	80	100	0,26	0,33
USB2TTL	CH340C	1	3,3	7	20	0,02	0,14
Red LED	0805 SMD	1	3,3	1,56	42	0,01	0,14
Blue LED	0805SMD	3	3,3	5,40	258	0,02	0,85
White LED	0805 SMD	3	3,3	4,08	285	0,01	0,94
GREEN LED	0603 SMD	1	3,3	1,00	1,13	0,00	0,004
Red LED	0603 SMD	1	3,3	0,76	0,83	2,49E-03	2,74E-03
DC Current Measurement	INA219A	1	3,3	10	10	0,033	0,033

CURRENT CONSUMPTION	TYP (A)	MAX (A)
AP3429/A_CURRENT-CONSUMPTION_MB	0,80	0,02
LM3671_CURRENT-CONSUMPTION_MB	0,50	0,06
AP3429/A_CURRENTCONSUMPTION_BB	0,74	0,02
LM3671_CURRENT-CONSUMPTION_BB	0,44	0,06

POWER CONSUMPTION	TYP (W)	MAX (W)
AP3429/A_POWER-CONSUMPTION_MB	2,64	-0,05
LM3671_POWER-CONSUMPTION_MB	1,65	0,94
AP3429/A_POWER-CONSUMPTION_BB	2,45	0,02
LM3671_POWER-CONSUMPTION_BB	1,01	1,0080469

CURRENT CONSUMPTION BUDGET

Title: Current Consumption		
Size: A4	Revision: 4	Date: 18/08/2021
Author: Irene Gil Martín		Sheet: 20 of 20

Appendix C

Simulations

C.1 Diode Simulation

In order to make a good design of the circuit to include the Light-Emitting Diode (LED) diodes we decided to simulate the circuit to make sure the calculation was correctly made. To do so, we have followed the tutorial given by Robert Bolanos, a Principal Research Engineer at Southwest Research Institute. [76]

Since we did not have the exact datasheet for the LEDs that we are using given that they were part of the laboratory stock, we decided to **model the LED diode** to be able to make a good calculation of the necessary resistances for its perfect operation.

We are going to be using a 0805 LED in three different colors: WHITE, BLUE and RED. As the characterization is the same for the three colors, we are going to proceed to explain only the BLUE LED although we have done the three characterizations.

Modeling a diode in any simulator such as *Pspice* or *Altium* requires knowing three basic parameters:

- **Emission Coefficient (N)** : Accounts for the recombination of holes with free electrons in the depletion region of the diode.
- **Reverse Saturation Current (IS)** : Is the part of the reverse current in a semiconductor diode caused by the diffusion of minority carriers from the neutral regions to the depletion region.
- **Series resistance (RS)** : This parameter is responsible for the curving of the current when entering the saturation zone.

With this three parameters, we would get an good modeling of the LED. If we wanted to make it more accurate we would have to add some parameters of the fabrication such as the activation energy (EG), the flicker noise coefficient and noise exponent, etc.

As we already mention, we do not have the datasheet for the LED so we have to get this three parameters through an experimental curve. In order to do so, we solder two wires at each extreme of the LED so that we can connect the multimeter and the voltage source as shown in Figure C.1. As the LED had a small package (0805) we decided to solder two wires to each side : one of enameled copper wire [1] directly to the LED, and other standard wire to the enameled copper wire.

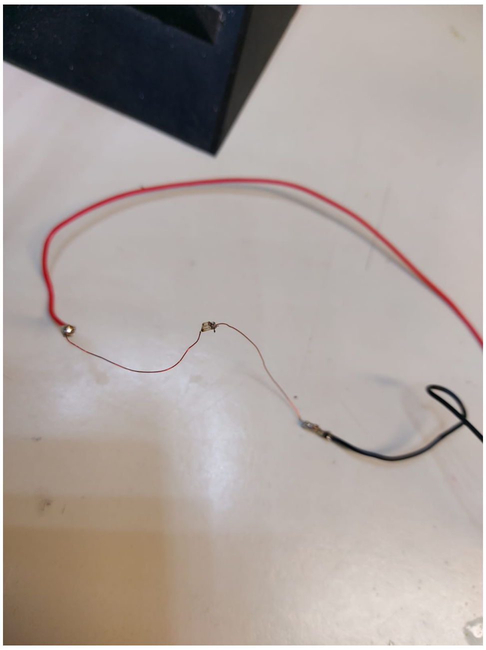


Figure C.1 – *Wires Solder on a* 0805 *BLUE LED*

Then, we have measured the current through the LED for drawing the $I_{DS} - V_T$ curve. As we were measuring current, we connected the multimeter and the voltage source on an open circuit (Figure C.2).

[1] single-wire cable that, through enamel, prevents possible short circuits. To remove the enamel to weld it, all you have to do is apply heat with the welder to the section to be welded. As they have such a small section, you have to be very careful with their length as it can raise the resistance of the cable to non-negligible values ($R = \rho \frac{L}{s}$)

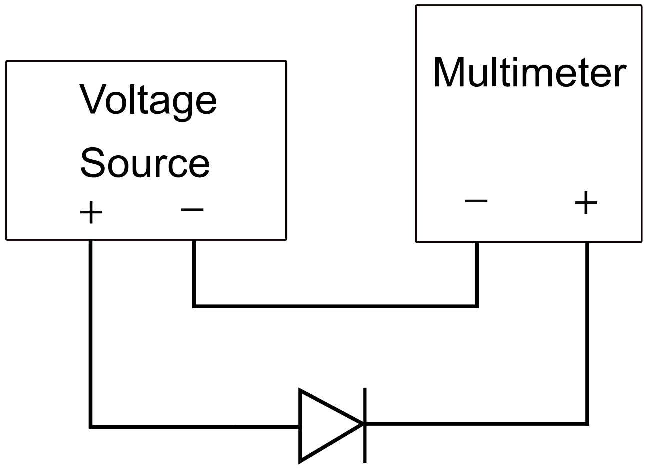


Figure C.2 – *Measuring current with an open circuit*

Implementing that circuit, we were able to obtain the experimental curve shown on Figure C.3 by plotting the values on *Python*.

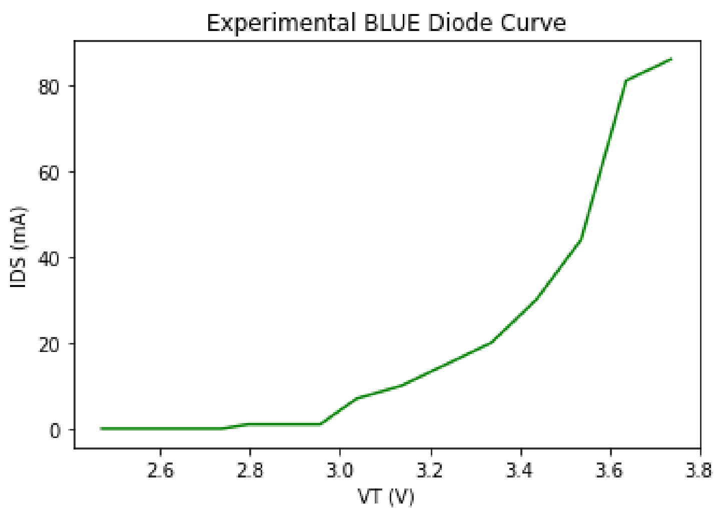


Figure C.3 – *Experimental BLUE Diode LED curve*

Once we had the experimental curve, the equation needed for obtaining IS, N, RS is (C.1.1). With this equation and implementing a *Newton-Raphson* iteration method [2] we were able to obtain this three parameters by implementing a small script (**??**) on *Python*.

$$I_S = R_S\, I_{DS} \;+\; N\; VTH\; ln\,(\frac{I_{DS}}{IS}) \tag{C.1.1}$$

[2] Method that allows finding a root of a non-linear equation as long as one starts from a good initial estimate of it

```
"""IMPORT LIBRARIES"""

import matplotlib.pyplot as plt
from numpy import *
from scipy.optimize import *
import math
import numpy as np
import sympy as sp

"""DEFINING DATA"""

VD_B=[2.47,2.536,2.666,2.736,2.796,2.816,2.956,3.037,3.137,3.237,3.337,3.437,3.537,3.637,3.737];
ID_B=[0.0557161e-3,1.33E-04,0.463883e-3,0.7279e-3,1.007007e-3,1.081657e-3,1.798637e-3,7.40465e-3,10.8335e
  -3,15.3832e-3,20.82126e-3,30.4364e-3,44.9937e-3,81.41153e-3,86.5511e-3];
ID_B_mA = np.array([1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]);
for i in range (15):
    ID_B_mA[i] = ID_B[i]*1000;

SIZE_ID_B=len(ID_B);

VTH=25.8e-3; #THERMAL VOLTAGE

#Iterating for only some values

"BLUE LED"
VD1_B=VD_B[8];
VD2_B=VD_B[9];
VD3_B=VD_B[14];

ID1_B=ID_B[8];
ID2_B=ID_B[9];
ID3_B=ID_B[14];


"FUNCTION RESOLUTION"
x_initguess=[9e-8,0.01,4]

"BLUE LED"

def func(diode_B): #Defining the function
    IS_B,RS_B,N_B = diode_B #Defining the unknown parameters
    return[RS_B * ID1_B + N_B * VTH *np.log(ID1_B/IS_B)-VD1_B, #Implementing the equations
           RS_B * ID2_B + N_B * VTH *np.log(ID2_B/IS_B)-VD2_B,
           RS_B * ID3_B + N_B * VTH *np.log(ID3_B/IS_B)-VD3_B]
          # RS * ID4 + N * VTH *np.log(ID4/IS)-VD4]

def jacobian(diode_B): #Defininh the funtion jacobian
    IS_B,RS_B,N_B = diode_B
    return [[N_B*VTH*(-1/IS_B),ID1_B,VTH*np.log(ID1_B/IS_B)],
           [N_B*VTH*(-1/IS_B),ID2_B,VTH*np.log(ID2_B/IS_B)],
           [N_B*VTH*(-1/IS_B),ID3_B,VTH*np.log(ID3_B/IS_B)]]
          #[N*VTH*(-1/IS),ID4,VTH*np.log(ID4/IS)]]

def newton_iter(fun,x_init,jaco): #Starting the iteration for solving the equations
    max_iter = 100
    epsilon = 1e-5

    x_last = x_init
    for k in range(max_iter):
        J=np.array(jaco(x_last))
        F=np.array(func(x_last))
        delta=np.linalg.solve(J,-F)
        x_last=x_last+delta
        if np.linalg.norm(delta)<epsilon:
          print('converges at k=',k)
          break
        else:
            print('no solution found')
        return x_last

x_solution_B=newton_iter(func,x_initguess,jacobian) #Solution with the Newton-Raphson method


"PRINTING THE SOLUTIONS ON THE CONSOLE"

print()
print('There are:',SIZE_ID_B,'current samples of the BLUE LED')
print()
print('BLUE Diode parametres:')
print('IS:',x_solution_B[0])
print('RS:',x_solution_B[1])
print('N:',x_solution_B[2])
print()

"PLOTTING THE EXPERIMENTAL CURVES"
```

```
plt.plot(VD_B,ID_B_mA,'g')
plt.title('Experimental BLUE Diode Curve')
plt.ylabel('IDS (mA)')
plt.xlabel('VT (V)')
plt.show()
```

Code C.1 – *Python Script for modeling the BLUE LED*

The values obtained after running the scripts were:

$$I_S = 2.453679415712279\ e^{-07}$$
$$N = 10.978583532052244$$
$$R_S = 0.15037170339351943$$

Once we had the three parameters, we were able to run the simulation on Altium obtaining the following curve:

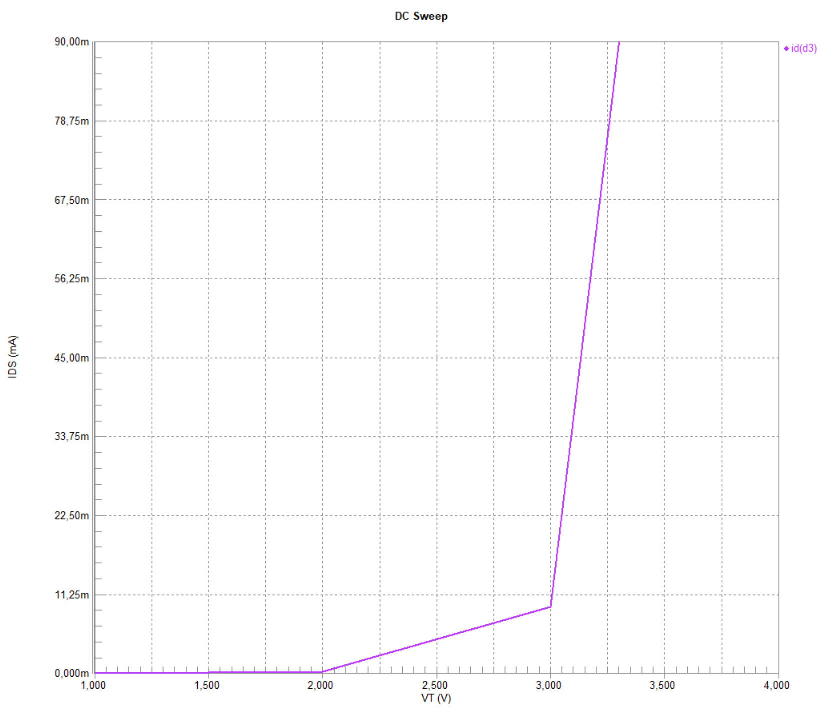


Figure C.4 – *Blue diode's ideal curve simulated on Altium*

Comparing the real curve obtained by measuring the LED on the laboratory Figure C.3 with the ideal curve obtained on Altium Simulator Figure C.4 we can assure that even though the values are similar, they are not the same. This is due to the characterization of a LED with only these three parameters is something complex that only allows us to obtain approximate values with a higher or lower current, but not an exact curve. In fact, the characterization we have obtained is a high current characterization. This can be observed by noticing that when the current rapidly increases, the values obtained on the ideal curve are much higher than those obtained in the laboratory.

C.2 Antennas Frequency Measurement

For both the radio module and GPS we need antennas that will allow their correct operation. Since we have studied the Telecommunication degree, we should be able to analyze the RF parameters.

On this appendix section we will be presenting the method we use for measuring some of this parameters such as the frequency, the impedance or the dB. For measuring this parameters we will be using a Vector Network Analyzer (VNA) that was available on the laboratory (Figure C.5) [59].

(a) *Device OFF* **(b)** *Device OFF*

Figure C.5 – *VNA available on the laboratory*

The VNA is a device that will allow us to measure RF parameters of the device under test. Not only does the it measure the amplitude response, but it also looks at the phase as well. As a result VNA network analyzer may also be called a gain-phase meter or an Automatic Network Analyzer.

For using this device we will be using his associated software [59]. This software will allow us to visualize all this parameters on our PC.

For explaining the procedure we have used for measuring different antennas, we will be using the GPS antenna we used for the Open CanSat Base Board. Since they did not include one on their kit, we took one of the antennas available on the laboratory. Consequently, we needed to assure this antenna was the right one for our device.

The first thing we need to do is to interface the VNA with the computer via USB, connect the antenna to the double side SMA connector and open the software. Once we opened the software, we needed to select the COM port and wait until the data is synchronized.

Then, for performing an accurate measure, we need to calibrate the device. Depending on the device we are measuring, we should select the frequency window so that the data is more accurate. In our case, the antenna is a GPS antenna, therefore its frequency is 1575.42 MHz. We will be selecting a frequency window of 1400 - 1600 MHz with 300 points (0.67 MHz step size). For calibrating the device we need to connect the different calibration connector (short-circuit,open-circuit and matching impedance (50 Ω)) shown in Figure C.6 and press the corresponding button for the VNA to lecture that measure. When we have performed the three different measures, our VNA is ready to measure any device within this frequency (Figure C.7).

(a) *Top View* **(b)** *Bottom View*

Figure C.6 – *Calibration connectors (from left to right: short-circuit, open-circuit and matching impedance)*

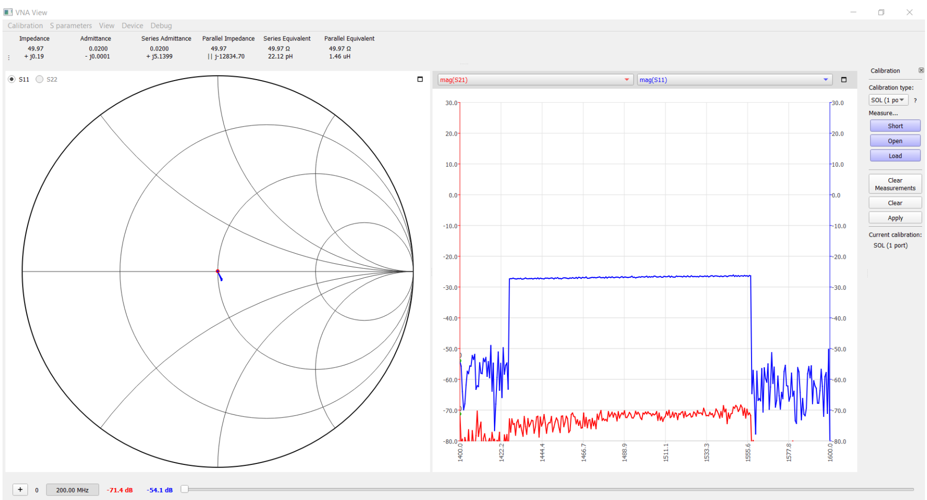


Figure C.7 – *Capture from the VNA with the matching impedance connector after calibration*

Finally, we will connect the antenna we selected from the laboratory and capture the measure. This measure is the one exhibit in Figure C.8.

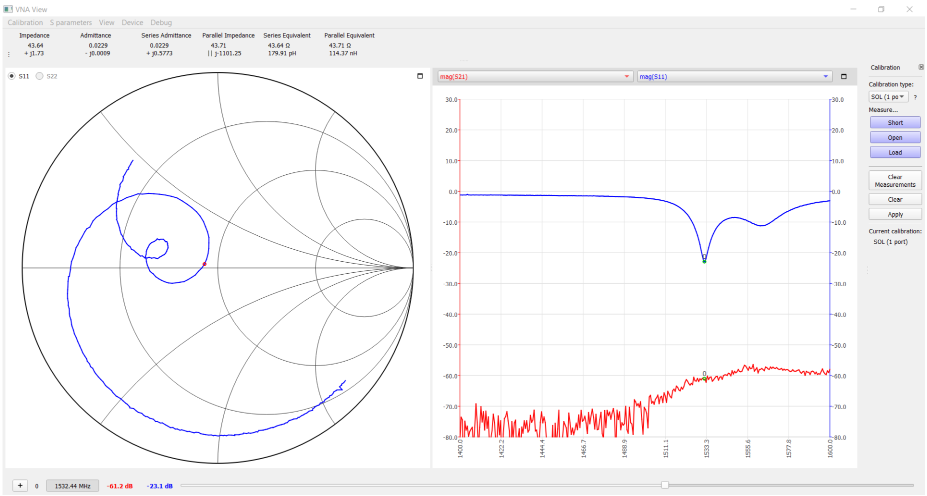


Figure C.8 – *Capture from the VNA software when measuring the GPS antenna*

We can observe from this picture that the frequency of operation of this antenna is 1532.4 MHz. We can also observe that the impedance of this antenna is 50 Ω and -23.1 dB gain on this frequency. Besides, on the frequency of interest, 1575.42 MHz the gain this antenna presents is −7.2 dB.

We also tested the radio antenna RF parameters. We also calibrate the device but in a different frequency windoe (200 - 1000 MHz) and connected the antenna. The results are presented in Figure C.9.

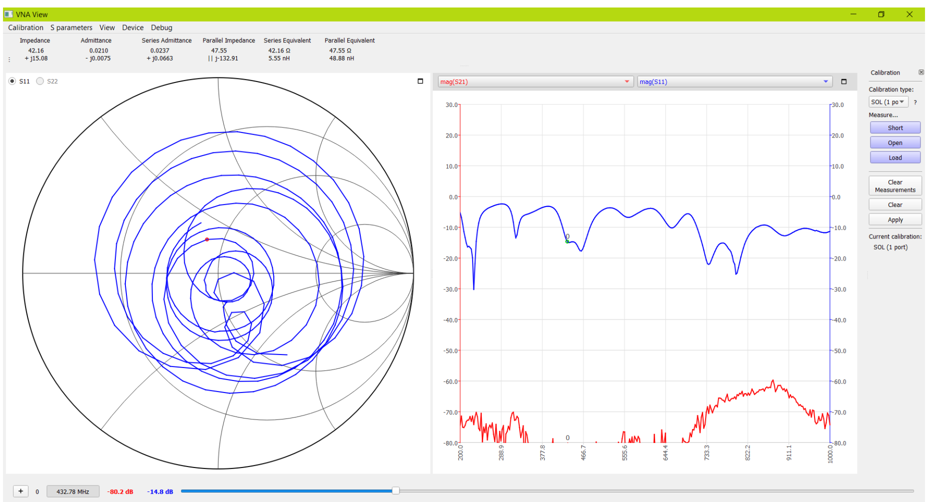


Figure C.9 – *Capture from the VNA software when measuring the radio antenna*

We can observe that at the frequency of interest, 433 MHz, the gain this antenna presents is −20.8 dB. We can also see that this antenna is a 50 Ω antenna.

Appendix D

Proper Debugging

D.1 Restoring Bootloader

In this section we will be explaining how to restore the bootloader in case we somehow damage it. As we are programming directly into the microprocessor, this is a fairly frequent situation that we might face.

First of all we should start talking about the bootloader. As we have already defined in this thesis glossary, the bootloader is a *.hex* file (small program) written into the flash memory of the microprocessor which allows code and EEPROM data to be transmitted over a serial cable and written to the microprocessor [35]. This file has to be written only once (ideally) into the flash memory using a conventional programmer. However, if we somehow commit a mistake while programming the file and erase or overwrite any file of the flash memory, this file would be damaged and we will lose the serial communication.

At this point we may wonder how could we restore the serial communication if we have already lost it. Depending on the microprocessor we are using, we will need a in-circuit debugger [44] that will let us to communicate with the microprocessor and re-write this file. An in-circuit debugger is, in simple words, a device that translate the commands send from the PC (serial communication) to language (SWD or JTAG protocol) that is understood by the microprocessor [43]. Not every microprocessor can use the same in-circuit debugger, therefore, before thinking of designing any device with a microprocessor, we will need to research which is the one we need in case something goes wrong with the bootloader.

Given that in this Bachelor's Thesis we will be working with two different microprocessor we will review the process for restoring both bootloaders. It should be noted that in Arduino IDE, restoring the bootloader of the microprocessor breakout boards is much more simpler than restoring it when we do not have this breakout boards.

D.1.1 SAMD21G18A-AUT Bootloader's Restoration

To communicate with the board when the bootloader has been damaged on this microprocessor, we will need to establish a SWD communication. As we are focusing on the microprocessor the Czech engineers have included on their device, we will also be using the particular bootloader they developed. Not every SAMD21G18A-AUT microprocessor need the same commands. It will depend on the device's application.

For establishing this SWD communication we will be needing some extra devices. Once we have encountered this needed material, we will be ready to restore the bootloader. The steps we will follow are the one listed below:

- Establishing the SWD communication between the microprocessor and the J-Link
- Opening the **ATMEL Studio** software for erasing the corrupted bootloader
- Uploading the uncorrupted bootloader and checking whether we can establish the serial communication or not

D.1.1.1 Materials required

The indispensable material to fulfill this process has been presented below:

- Open CanSat board with the corrupted bootloader
- *ATMEL Studio* software
- Bootloader for the Open CanSat microprocessor
- J-Link in-circuit debugger Device (Figure D.1)
- Some wires and male pin headers for connecting the Open CanSat to the J-Link

Figure D.1 – *J-Link Device from SEGGER [54]*

D.1.1.2 Corrupted Bootloader Verification

Before restoring the bootloader we will need to check if this file is indeed corrupted. To verify if the program we have uploaded into the board spoiled the bootloader we simply have to observe if our device is recognized as a COM port. If the device is not recognized as COM port, the bootloader must be restored. Furthermore, in some cases, even when the board is recognized as a COM port, the bootloader could be damaged. In this case, we will not be able to upload any program and we will received the following errors:

```
avrdue: ser_rcv(): programmer is not responding
avrdue: stk500v2_ReciveMessage(): timeout
```

Figure D.2 – *Corrupted Bootloader error*

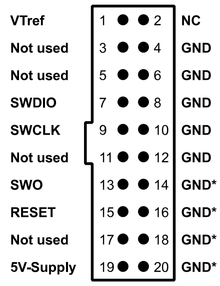
Once we have demonstrated that the file is corrupted, we will need to go on to establishing the SWD communication.

D.1.1.3 SWD Communication Establishment

In paragraph 2.1.1.1.9 we talked about the fact that this implemented this pads for setting up the interface. Therefore we will only need to solder a cable on each pad and to a male pin header for connecting the J-Link with the board. The connection we will need to make are the ones presented in Figure D.3.

PAD OPEN CANSAT	PIN RIBBON CABLE J-LINKER
3V3	VTref (1)
SWC	SWCLK (9)
SWD	SWDIO (7)
REST	RESET (15)
GND	GND (8*)

(a) *Table Format Connection*



(b) *Ribbon Connection*

Figure D.3 – *SWD J-Link interface [74]*

The connection of the CanSat transmitter to the Ribbon Cable of the J-Link would be like presented in Figure D.4.

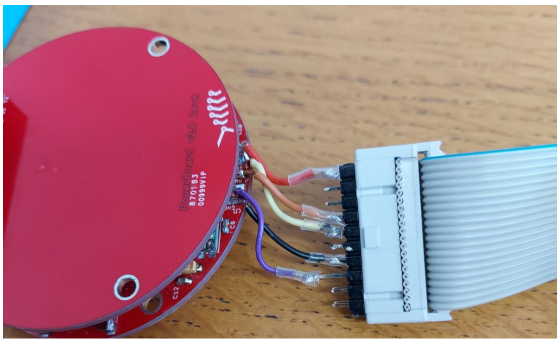


Figure D.4 – *CanSat transmitter connected to the SEGGER J-Link*

D.1.1.4 Bootloader Restoration

Once the board is connected to the J-Link, we will need to connect the J-Link to the PC. With this system, we can communicate from the PC with the J-Link with the serial communication and the J-Link is able to tell the board whatever we need with the SWD interface. For doing all this communication, we need to open the **ATMEL Studio**. Once the software is opened we need to unfold the*Programming Device* tab under the *Tool* tab. The window that will appear is the one presented in Figure D.5.

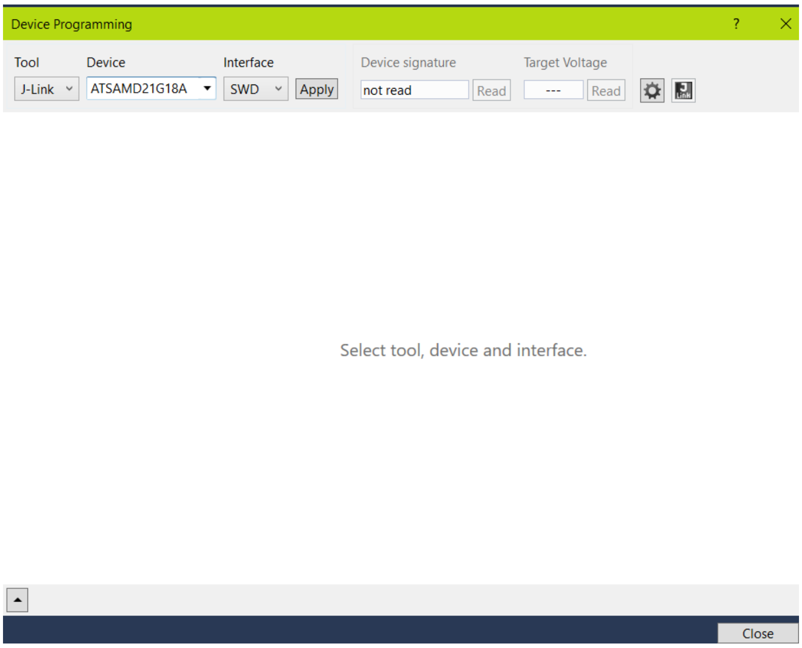


Figure D.5 – *Programming Device Window in ATMEL Studio*

The next thing we need to do is to power up the board and to pres the button for recognizing the microprocessor of the board. When this button is pressed, it also recognized the target voltage. The window we will be seeing once the microprocessor is recognized is the one presented in Figure D.6.

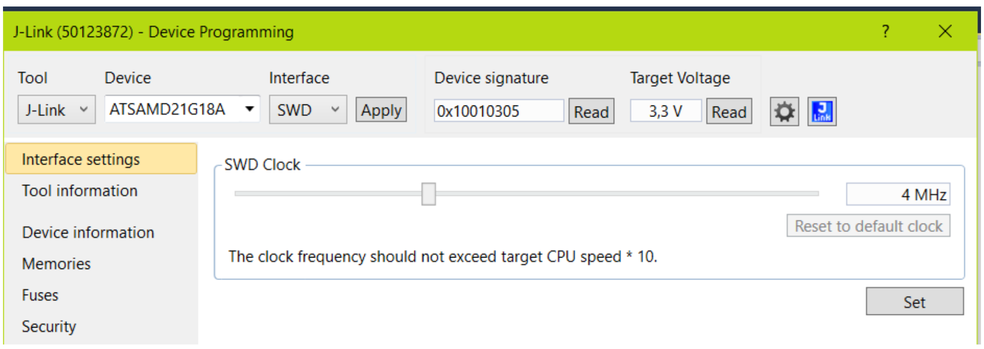


Figure D.6 – *Programming Device Window in ATMEL Studio with microprocessor recognition*

Now it is time for erasing the corrupted bootloader for being able to replace it with the uncorrupted one. This is done by clicking in the *Memories* menu and pressing the button erase now.

Once the corrupted bootloader is erased, we will go to the *Fuses* menu for checking whether we can communicate with the board or not before uploading the uncorrupted bootloader. Before trying the connection we need to check the following registers:

USER_WORD_0.NVMCTRL_BOOTPROT

USER_WORD1.NVMCTRL_REGION_LOOKS

The first register must be set to 0 whereas the second one must be set to 0xFFF. The datasheet [28] explicitly remark that this registers are the one that will let us program the microprocessor. If the values of those registers are not set correctly, we will not be able to write into the microprocessor memory Flash.

Finally, when the registers are set correctly, we need to upload the uncorrupted bootloader. This file is uploaded in the *Memories* menu, on the *Flash* configuration. There we will need to search for the file on our computer (we downloaded it from the Czech engineers git [3]) and press *program*.

D.1.2 ESP32-WROOM-32D Bootloader Reestablishment

Before explaining the steps we have to follow for reestablishing the serial communication with the ESP32-WROOM-32D microprocessor, we must remark that we have restored the bootloader in the Open CanSat kit since the microprocessor they used is less protected and allow the programmer to access the memory Flash registers.

However, we researched how to restore the bootloader in the ESP32-WROOM-32D in case this strange situation happens. It is indeed a strange situation since this microprocessor do not let the user to access the memory flash and write on it by accident.

The process that we will need to follow for restoring the bootloader requires the same materials we listed in subsubsection D.1.1.1. Nevertheless, instead of establishing a SWD communication we will be establishing a JTAG interface between the J-Link and the board. The differences between the SWD and JTAG standard are listed in Table D.1.2.

	SWD Standard	**JTAG Standard**
Pin Count	2	4
Supported CPU architectures	Only ARM	JTAG is an independent group and hence is supported by many architectures not just limited to ARM
Topology	Star	Daisy chained
Special Features	Printing debug info via debug port	Not supported

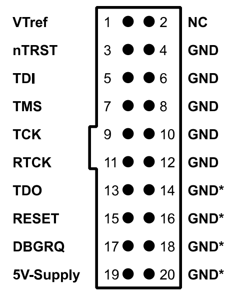
Table D.1 – *SWD vs JTAG Standard [43]*

D.1.2.1 JTAG Communication Establishment

The ESP32-WROOM-32D also uses a J-Link as a in-circuit debugger, but, as we already mentioned, the interface is with the JTAG standard. Therefore, the pin connection we will need to make for interfacing the microprocessor with the is the one presented in Figure D.7.

PIN ESP32-WROOM-32D	PIN RIBBON CABLE J-LINKER
GPIO12	JTAG TDI (5)
GPIO13	JTAG TCK (9)
GPIO14	JTAG TMS (7)
GPIO15	JTAG TDO (13)
RST/EN	JTAG Reset (15)
+3V3	VTref (1)
GND	GND

(a) *Table Format Connection*



(b) *Ribbon Connection*

Figure D.7 – *JTAG J-Link interface [74]*

D.1.2.2 Bootloader Restoration

The software needed for restoring the bootloader ***SysProgs USB Driver Tool*** [55]. Once we have downloaded and installed this software, we will only need to follow the steps detailed in the following link [67] for uploading the uncorrupted bootloader to the microprocessor.

Appendix E

Data Analysis

E.1 GPS Strings

The GPS modules are in charge of communicating with the satellites that are available in their position. Once the communication is established, this modules ask the available satellites for the information related to their location. This information is transmitted via **NMEA Data Strings**. NMEA-formatted GPS data makes life so much simple since a wide variety of GPS receivers support the same software and we do not have to write a custom interface for each one [50]. All GPS modules in the entire world communicates with the NMEA 0183 standard.

The NMEA Data Strings include data such as the position, speed or time of the device separated by commas. Each string of the 11 different each device can received begin with a $. After this dollar sign we will find the name of the string we are receiving (this name will tell us the information we will be receiving). Lastly, all strings finalizes with a checksum to check the correct receiving. .

There are 11 different sentences we can received from a satellites, each of one give us different information. This sentences are written in ASCII and differentiated with the header which is always preceded with a $. This sentences include information such as the latitude, longitude, time, etc separated by commas.

For receiving the NMEA strings we only need a GPS module powered with its needed DC power supply. Once the module is powered, it will start outputting data as soon as it has identified any GPS satellite within his range.

This sentences we could be receiving are the one listed below. Note that it can be arranged via software to only received certain strings.

- **GPALM:** GPS almanac data

- **GPGGA:** GPS fix data
- **GPGLL:** GPS antenna position data
- **GPGRS:** GPS range residuals
- **GPGSA:** GPS DOP (dilution of precision) and active satellites
- **GPGST:** GPS pseudorange statistics
- **GPGSV:** GPS satellites in view
- **GPMSS:** Beacon receiver signal status
- **GPRMC:** Recommended minimum specific GPS data
- **GPVTG:** Course over ground and ground speed
- **GPZDA:** GPS time and date

As we have already said, each of this string contains a different data information. As for us, the most important data and the ones we will be analyzing are the GPGGA and GPRMC data string. This strings both contains the information related tu longitude, latitude, speed, etc which will be interesting to review for our device operation.

The last thing we need to remark before analyzing the strings is the message ID. If we are receiving a string from a GPS, the message ID is the one we listed above. However, if our device is a GLONASS (aka GNSS) device, the message ID will be $**GN**GGA. Therefore, the two first letters of this message ID indicates the device we are using [51].

E.1.1 GPGGA NMEA string

For explaining the information that this string contains, its better if we analyze a received string of the one we received while developing the CanSat firmware. The string we will be analyzing is:

$GNGGA, 132207.00, 3704.48626, N, 00336.06479, W, 1, 07, 3.07, 869.3, M, 46.7, M, , *53

Field	Description	Example
Message ID	GGA protocol header	$GPGGA
UTC time	hhmmss.ss	13 : 22 : 07.00
Latitude	ddmm.mmmm	37º 04.48626′
N/S Indicator	N = North, S = South	N
Longitude	dddmm.mmmm	00º 336.06479′
E/W indicator	E = East or W = West	W

Continued on next page

Field	Description	Example
Position Fix Indicator	0 = Fix not available or invalid 1 = fix valid 2 = Differential GPS (DGPS),fix valid 3 - 5 = Not Supported 6 = Dead Reckoning Mode (DR), fix valid	1
Satellites used	Range is 0 to 12	7
HDOP	Horizontal Dilution of Precision	3.07
MSL Altitude	Meters	869.3
Units	Meters	M
Geoid Separation	Meters	
Units	Meters	M
Age of diff. corr.	Second	
Diff.ref. station ID		
Checksum	Bytes transmitted	*53

Table E.1 – *GPGGA NMEA string [52]*

Note that this string will give us not only the information about the time and space position but also the altitude of our device which will be important for comparing this data with the altitude measured with the BME280.

E.1.2 GPRMC NMEA string

Analogue to what we did for the GPGGA string in the last section, in this section will be analyzing a GPRMC string. To do such a thing we will analyze one of the strings we received during the test we perform through out this Bachelor Thesis. The string we will be reviewing is:

GNRMC, 132142.00, A, 3704.48595, N, 00336.06619, W, 0.236, , 050821, , , A*79

Field	Description	Example
Message ID	GGA protocol header	$GPGGA
UTC time	hhmmss.ss	13 : 21 : 42.00
Status	A = Data Valid V = Data not Valid	A
Latitude	ddmm.mmmm	37° 04.48595′
N/S Indicator	N = North, S = South	N
Longitude	dddmm.mmmm	00° 336.06619′

Continued on next page

Field	Description	Example
E/W indicator	E = East or W = West	W
Speed over the ground	knots	0.236
Course over the ground	degrees	
Date	ddmmyy	05/08/21
Magnetic variation	Degrees(E = East, W = West)	
Mode	A = Autonomous D = DGPS E = DR	A
Checksum	Bytes transmitted	*79

Table E.2 – *GPRMC NMEA string [52]*