UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

BARCELONA SCHOOL OF INFORMATICS
BACHELOR'S DEGREE IN INFORMATICS ENGINEERING
MAJOR IN COMPUTER ENGINEERING

Design and Implementation of a 433MHz RF IoT Gateway

BACHELOR'S THESIS

Alex Rodríguez Navas

January 2023

Thesis Director
Enric Xavier Martin Rull

GEP Tutor
Paola Lorenza Pinto

This page is intentionally left blank.

Acknowledgements

I am incredibly grateful to my Thesis director, Enric Xavier, for his guidance and support throughout the process of completing my bachelor's Thesis. His expertise and insights were invaluable, and I feel fortunate to have had such a knowledgeable and dedicated mentor.

I would also like to express my sincere appreciation to my GEP tutor, Paola Lorenza, for her encouragement and aid on the project management phase enabling me to prepare my Thesis successfully.

I am deeply thankful to Carlos Morata for his invaluable and crucial assistance during the completion of a critical part of the project.

I also would like to extend my thanks to the UPC university for providing me with the opportunity to pursue my education and for the resources and support they have provided throughout my time here.

Finally, I am also extremely thankful to my family for their love, support, and encouragement as I completed my bachelor's degree. Their belief in me and my abilities has been a constant source of motivation, and I couldn't have done it without them.

Abstract

The growth of IoT technology has been apparent over the past decade, however only 0.06% of devices that could be connected to the Internet actually are. Every second 200 new IoT devices connect to the Internet for the first time. This number is expected to increase to a new whole order of magnitude by 2025 generating up to 11 trillion dollars in economic value.

Meanwhile, right now in our houses, there exists a huge portion of those not-so-old dorky devices that can be smartified, enabling a new set of capabilities by hooking them up to the Internet. 433MHz sensors and actuators span through a variety of categories, from weather stations or motion detection alarms, to wireless switches or motorized shutters.

This Bachelor's Thesis carries out the design and the creation of a secure, Internet-connected gateway which brings Internet of Things capabilities for the widely used 433MHz sensors and actuators band.

This final degree project is not to just design or build a bare-simple hardware prototype, but rather taking a step above, contemplating additional aspects of a retail electronic device such as manufacturing the PCB, the casing and even the packaging. The final outcome is a very professional product prototype that could potentially be bought in a general consumer electronics shop.

The final user of the device can connect her/his 433MHz sensors and actuators to the gateway, receive and transmit data or schedule commands to them, securely, reliably and through the Internet.

Resumen

El crecimiento de la tecnología IoT ha sido evidente en la última década, sin embargo, solo el 0,06% de los dispositivos que podrían estar conectados a Internet lo están. Cada segundo, 200 nuevos dispositivos IoT se conectan a Internet por primera vez. Se espera que este número aumente a una nueva orden de magnitud para 2025, generando hasta 11 millones de billones de dólares en valor económico.

Mientras tanto, en nuestras casas, existe una gran cantidad de esos dispositivos no tan viejos y tontos que pueden llegar a ser inteligentes, habilitándoles un nuevo conjunto de capacidades al conectarlos a Internet. Los sensores y actuadores de 433MHz abarcan una gran variedad de categorías, desde estaciones meteorológicas o alarmas de apertura, hasta interruptores inalámbricos o persianas motorizadas.

Este proyecto final de grado lleva a cabo el diseño y la creación de un gateway seguro y conectado a Internet que trae capacidades IoT para la banda de sensores y actuadores ampliamente utilizada de 433 MHz.

Este proyecto final de grado no se trata solo de diseñar o construir un prototipo de hardware simple, sino dar un paso más allá, contemplando aspectos adicionales de un producto electrónico, como la fabricación del PCB, la carcasa e incluso la caja. El resultado final es un prototipo de producto muy profesional que potencialmente podría comprarse en una tienda de electrónica de consumo.

El usuario final del dispositivo puede conectar sus sensores y actuadores de 433MHz al gateway, recibir y transmitir datos o programarles comandos, de manera segura, fiable y a través de Internet.

Resum

El creixement de la tecnologia IoT ha estat evident en l'última dècada, però només el 0,06% dels dispositius que podrien estar connectats a Internet ho estan. Cada segon, 200 nous dispositius IoT es connecten a Internet per primer cop. S'espera que aquest nombre augmenti a una nova ordre de magnitud per al 2025, generant fins a 11 milions de bilions de dòlars en valor econòmic.

Mentrestant, a casa nostra, hi ha una gran quantitat d'aquests dispositius no tan vells i ximples que poden arribar a ser intel·ligents, habilitant-los un nou conjunt de capacitats al connectar-los a Internet. Els sensors i els actuadors de 433MHz abasten una gran varietat de categories, des d'estacions meteorològiques o alarmes d'obertura, fins a interruptors sense fil o persianes motoritzades.

Aquest projecte final de grau duu a terme el disseny i creació d'un gateway segur i connectat a Internet que porta capacitats IoT per a la banda de sensors i actuadors àmpliament utilitzada de 433MHz.

Aquest projecte final de grau no es tracta només de dissenyar o construir un prototip de hardware simple, sinó d'anar un pas més enllà, contemplant aspectes addicionals d'un producte electrònic, com ara la fabricació del PCB, la carcassa i fins i tot la caixa. El resultat final és un prototip de producte molt professional que potencialment es podria comprar a una botiga d'electrònica de consum.

L'usuari final del dispositiu pot connectar els seus sensors i actuadors de 433MHz al gateway, rebre i transmetre dades o programar ordres, de manera segura, fiable i a través d'Internet.

Table of Contents

1. Context and Scope **18**
1.1. Introduction and contextualization 18
Context 18
Concepts 20
Problem definition 21
Stakeholders 21
1.2. Justification 22
Existing solutions 22
Value proposal 23
1.3. Scope 24
Requirements 24
Objectives 26
Potential obstacles and risks 28
1.4. Methodology and rigor 29
Validation strategy 29
Developing, designing, analytical and monitoring tools 30

2. Project Planning **31**
2.1. Task list 31
Resources 31
Management tasks (MGT) 32
Firmware tasks (FWE) 33
Schematic tasks (SMC) 36
PCB tasks (PCB) 37
Casing tasks (CSG) 37
Packaging tasks (PKG) 38
Summary 39
2.2. Gantt chart 40
2.3. Risk management 41
Firmware risks 41
Hardware risks 42
Limiting risks 42
Personal risks 42
2.4. Changes 43
Revisited gantt chart 43

3. Budget **45**
3.1. Cost identification and estimation 45
Staff costs 45

Material costs 47
Generic costs 49
Incidental costs 49
Contingency costs 50
3.2. Final budget 50
3.3. Management control 51

4. Sustainability 52
4.1. Self assessment 52
4.2. Economic dimension 52
4.3. Environmental dimension 55
4.4. Social dimension 57

5. Laws and Regulations 59
5.1. Software 59
General data protection regulation 59
Cyber resilience act 59
5.2. Hardware 60
Restriction of hazardous substances 60
Electromagnetic compatibility 60
Radio equipment directive 60
Conformitè europëenne 60
Waste electrical and electronic equipment 60
General product safety directive 61
Common charger directive 61

6. Knowledge Integration 62
6.1. Informatics engineering degree 62
6.2. Computer engineering major 63
6.3. Technical competences 64

7. Initial Research 67
7.1. Test subjects 67
7.2. Communication protocol 68
Frequency analysis 68
Modulation analysis 69
Phase analysis 70
Data analysis 71
Results 74

8. Taken Approach 75
8.1. Proposed system 75
8.2. Decisions taken 76

8.3. Conceptual diagrams 77
Schematic 77
PCB 79
Casing 81
Packaging 82
Firmware 84

9. Hardware Components 98
9.1. Microcontroller 99
9.2. Receiver 100
9.3. Transmitter 100
9.4. Antennas 101
9.5. Real time clock 101
9.6. RGB LED 101
9.7. USB-C connector 102
9.8. Voltage regulator 103
9.9. Circuit protectors 104
Over-current 104
Over-voltage and reverse bias 104
Electrostatic discharge 104
9.10. Miscellany 105
Push buttons 105
Resistors 105
Capacitors 105
Inductors 105

10. Schematic 106
10.1. Design 106
Microcontroller 106
Receiver 109
Transmitter 111
Real time clock 112
RGB LED 113
USB-C connector and protection circuits 114
Voltage regulator 115
Mounting holes 116
10.2. Testing 117
10.3. Problems and solutions 118
10.4. Results 121

11. PCB 122
11.1. Design 122

Microcontroller 122
Receiver 124
Transmitter 127
Real time clock 128
RGB LED 129
USB-C connector and protection circuits 130
Voltage regulator 132
Mounting holes 132
3D preview 133
11.2. Manufacturing 134
Board 134
SMD components 135
THT components 137
11.3. Testing 138
11.4. Problems and solutions 139
11.5. Results 140

12. Casing 143
12.1. Design 143
Body 143
Lid 143
Snap fit closing 144
3D preview 145
12.2. Manufacturing 146
Slicing 146
Printing 147
Finishing 148
12.3. Testing 149
12.4. Problems and solutions 150
12.5. Results 151

13. Packaging 154
13.1. Design 154
Dieline 154
Top 154
Bottom 155
Front 156
Back 156
Left 156
Right 157
3D preview 158
13.2. Manufacturing 159

Package 159
Separator 160
13.3. Testing 161
13.4. Problems and solutions 162
13.5. Results 163

14. Firmware 166
14.1. Design 166
Architecture 166
Device definitions 167
Device receiver 169
Device transmitter 171
Wi-Fi provisioner 172
Status controller 173
Time synchronizer 173
Trigger scheduler 173
Frontend serving 174
CRUD serving 174
14.2. Testing 177
14.3. Problems and solutions 178
14.4. Results 178

15. Final Results 179
15.1. Demonstration 179
15.2. Achievement of goals 180

16. Conclusions 181
16.1. Summary and personal opinion 181
16.2. Future work 182

17. Bibliography 183

List of Figures

1. Number of global active IoT connections in billions by year **18**
2. Percentages of global searches about existing solutions on Google **22**
3. Gantt chart of the tasks **40**
4. Revisited Gantt chart of the tasks **44**
5. Splenssy open-close alarm **67**
6. SatChef wireless socket **67**
7. Master Otello wireless motorized blinds **67**
8. Radio frequency heatmap of the test subjects **68**
9. Scanned signals graphic of the wireless blinds **69**
10. Scanned signals graphic of the wireless blinds using ASK modulation **69**
11. Decomposition of the "up" command signal of the wireless blinds **70**
12. Mixed decomposition of two "up" command signals of the wireless blinds **70**
13. Data pattern decomposition of the "up" command signal of the wireless blinds **71**
14. Decoded bits of the "up" command signal of the wireless blinds **71**
15. Summarized communication protocol analysis of the open-close alarm **72**
16. Summarized communication protocol analysis of the wireless socket **73**
17. Conceptual diagram of the system **75**
18. Conceptual diagram of the schematic **87**
19. Conceptual diagram of the printed circuit board **79**
20. Conceptual diagram of the casing **81**
21. Conceptual diagram of the packaging **82**
22. Conceptual diagram of the backend firmware **84**
23. UML diagram of the backend models **86**
24. Cron expression syntax **88**
25. Conceptual diagram of the frontend dashboard page **94**
26. Conceptual diagram of the frontend devices, triggers, users and roles sections **95**
27. Conceptual diagram of the frontend system page **96**
28. Conceptual diagram of the frontend authentication page **97**
29. Conceptual diagram of the frontend onboarding page **97**
30. ESP32-S3-WROOM-2-N32R8V **99**
31. QAM-RX10-433 ..... **100**
32. QAM-TX2-433 ..... **100**

33. W3127 …… **101**
34. DS3231SN …….. **101**
35. CLX6F-FKC ……... **101**
36. BC847B …….. **102**
37. USB4085 …… **102**
38. LDL1117S33R …… **103**
39. MP008293 …….. **103**
40. MF-MSMF075/24 …….. **104**
41. 1N5819HW-7-F …….. **104**
42. USBLC6-2SC6 ……... **104**
43. KMR421G-LFS …….. **105**
44. Resistor sample …….. **105**
45. Capacitor sample …… **105**
46. Inductor sample …….. **105**
47. Microcontroller schematic part …….. **106**
48. ESP32-S3-WROOM-2-N32R8V pin list ……... **107**
49. ESP32-S3 series power on sequence and system reset …….. **108**
50. 433MHz receiver schematic part ……..... **109**
51. QAM-RX10-433 pin list …… **110**
52. Logic level standards diagram ……..... **110**
53. 433MHz transmitter schematic part ……... **111**
54. QAM-TX2-433 pin list ……... **111**
55. Real time clock schematic part ……... **112**
56. DS3231SN pin list …….. **112**
57. RGB LED schematic part ……... **113**
58. CLX6F-FKC pin list ……... **113**
59. USB-C connector and protection circuits schematic part …….. **114**
60. USB4085 pin list ……..... **114**
61. USBLC6-2SC6 pin list ……... **115**
62. 3.3V voltage regulator schematic part ……..... **116**
63. LDL1117S33R pin list ……..... **116**
64. M2 Mounting holes schematic part ……..... **116**
65. Testing of the 433MHz receiver schematic part …… **117**
66. Electrical rule check of the schematic ……..... **117**

67. Schematic part of the floating emitters problem ……... **118**
68. Schematic part of the microcontroller boot problem …….. **119**
69. Schematic part of the transmission antenna problem ……. **120**
70. Complete electrical schematic ……... **121**
71. Microcontroller PCB part ……..……. **122**
72. ESP32-S3-WROOM-2-N32R8V pin layout …….. **123**
73. Keepout zone for ESP32-S3 series module's antenna ……... **124**
74. 433MHz receiver PCB part …… **124**
75. QAM-RX10-433 pin layout ……... **125**
76. W3127 PCB layout ……. **125**
77. Aisler's 2-layer PCB parameters ……... **126**
78. KiCad's impedance matching calculator for the 433MHz antenna lines ……... **126**
79. 433MHz transmitter PCB part …… **127**
80. QAM-TX2-433 pin layout ……. **127**
81. Real time clock PCB part ……... **128**
82. DS3231SN pin layout ……. **128**
83. RGB LED PCB part ……... **129**
84. CLX6F-FKC pin layout ……. **129**
85. USB-C connector and protection circuits PCB part …….. **130**
86. USB4085 pin layout ……... **130**
87. USBLC6-2SC6 pin layout …….. **131**
88. KiCad's impedance matching calculator for the USB-C 2.0 data lines ……. **131**
89. 3.3V voltage regulator PCB part …… **132**
90. LDL1117S33R pin layout …….. **132**
91. M2 Mounting holes PCB part …… **132**
92. PCB 3D render top and bottom view ……. **133**
93. Aisler PCB order overview …… **134**
94. Aisler PCB preview top and bottom views …… **134**
95. Aisler PCB manufactured top and bottom views ……... **135**
96. SMD components placed with solder paste ……... **135**
97. Electronics oven used ……. **136**
98. ESP32-S3-WROOM-2 reflow profile and electronics oven configuration …… **136**
99. PCB with SMD components soldered top and bottom views …… **137**
100. PCB with THT components soldered top and bottom views ……... **137**

101. Design rule check of the PCB …….. **138**
102. Testing of the PCB ……... **138**
103. PCB part of the reversed transmitter problem …….. **139**
104. PCB part of the transmission antenna problem …… **139**
105. Complete PCB design …….. **140**
106. Complete PCB 3D render ……. **141**
107. Complete PCB manufactured ……... **142**
108. Body casing part ……... **143**
109. Lid casing part …….. **144**
110. Snap fit closing casing part ……... **144**
111. Casing 3D render top and bottom view …… **145**
112. Casing 3D render angle view ……...…… **145**
113. Casing slicing ……... **146**
114. Casing body printing …… **147**
115. Casing lid printing …… **147**
116. Casing body and lid manufactured ……... **148**
117. Casing RGB LED plastic lens …….. **148**
118. Casing and PCB simulation …….. **149**
119. Testing of the casing ……. **149**
120. Casing part of the tight snap fit closing problem …….……... **150**
121. Complete casing design …… **151**
122. Complete casing 3D render …….. **152**
123. Complete casing manufactured ……..... **153**
124. Packaging dieline ……. **154**
125. Top packaging part ……... **155**
126. Post-processing of the 3D model of the casing …… **155**
127. Bottom packaging part ……. **156**
128. Front packaging part ……. **156**
129. Back packaging part ……. **156**
130. Left packaging part ……... **157**
131. Right packaging part …… **157**
132. Packaging 3D render left and right side views ……. **158**
133. Packaging 3D render back view ……... **158**
134. Packly packaging order overview …… **159**

135. Packly packaging manufactured ……... **159**
136. Packaging bottom mount …….. **160**
137. Packaging hanging holes …….. **160**
138. Packaging polystyrene separator …….. **160**
139. Packaging dieline simulation ……... **161**
140. Testing of the packaging ……... **161**
141. Packaging setup instructions incongruence …….. **162**
142. Complete packaging design ……. **163**
143. Complete packaging 3D render …… **164**
144. Complete packaging manufactured …….. **165**
145. Backend firmware dependency graph …….. **167**
146. Device receiver algorithm …… **170**
147. Device transmitter algorithm …… **171**
148. Wi-Fi provisioner algorithm ……. **172**
149. Status controller algorithm ……... **173**
150. Create API endpoint algorithm …… **175**
151. Read API endpoint algorithm ……... **175**
152. Update API endpoint algorithm ……... **176**
153. Delete API endpoint algorithm …… **176**
154. End to end testing of the backend firmware ……. **177**
155. Linting of the backend firmware …….. **177**
156. USB-C JTAG debugging of the backend firmware …….. **177**
157. Complete backend firmware …… **178**
158. Complete Thesis project ……... **179**
159. Power consumption of the device …… **179**

List of Tables

1. Summary of the tasks ……. **39**
2. Average gross salaries in Spain …… **46**
3. Staff costs …… …… **47**
4. Material costs …… **48**
5. Generic costs …… **49**
6. Incidental costs …… **50**
7. Final budget …… **50**
8. Summary of the firmware tasks running on the operating system ……. **85**
9. Bill of materials, electrical characteristics and absolute maximum ratings ……... **98**
10. ESP32-S3-WROOM-2-N32R8V strapping pins …… **109**
11. Backend firmware project structure ……... **167**

1. Context and Scope

1.1. Introduction and contextualization

Context

Every second 200 new *IoT* devices connect to the Internet for the first time. This number is expected to increase to a new whole order of magnitude by 2025 generating up to 11 trillion dollars in economic value [1].

As seen in *Figure 1* the growth of IoT technology has been apparent over the past decade, however only 0.06% of devices that could be connected to the Internet actually are [1].

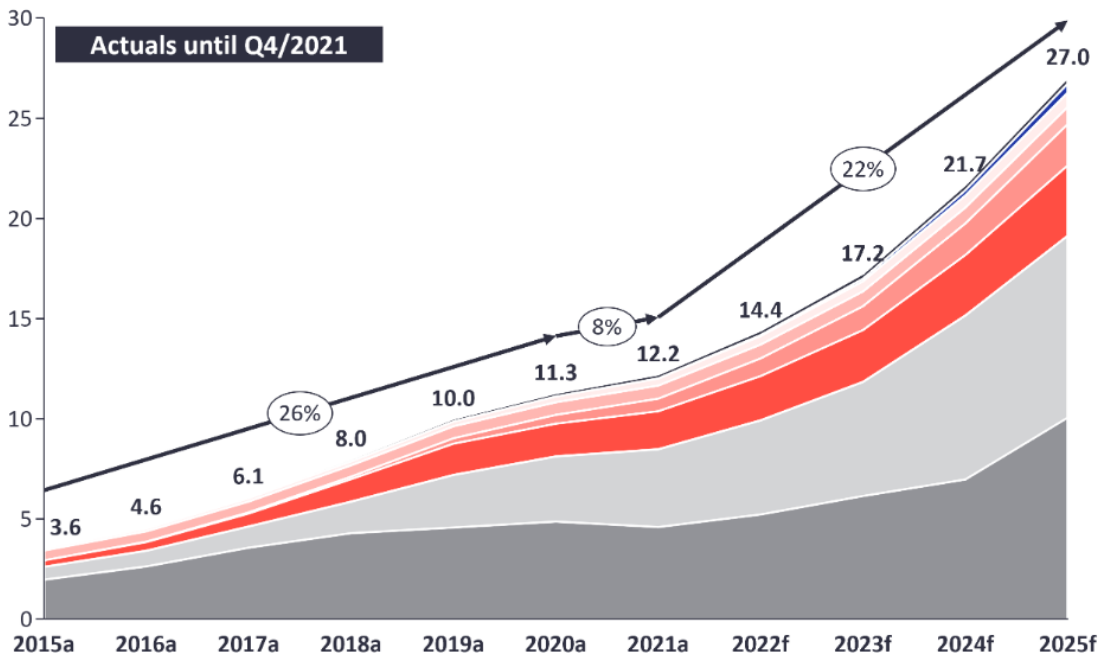


Figure 1: Number of global active IoT connections in billions by year.
Source: Mohammad Hasan [2].

Meanwhile, right now in our houses exists a huge portion of those not-so-old *dorky* devices that can be *smartified*, enabling a new set of capabilities by hooking them up to the Internet.

These single-purpose appliances span through a variety of categories such as sensors: weather stations, motion detection alarms, doorbells or thermometers; and actuators, including gauges, switches, garage doors or automated shutters.

The common denominator of these devices is the usage of the 433MHz radio frequency band for data communication, as it allows for a simple, cheap, low power consumption and license-free[1] design.

This A[2]-Modality Bachelor's Thesis of the Major in Computer Engineering of the Informatics Engineering Degree, done at the Barcelona School of Informatics and directed by the Vice-dean for Promotion and Communication Enric Xavier Martin Rull, carries out the design and creation of a secure, Internet-connected gateway which brings Internet of Things capabilities for the widely used 433MHz sensors and actuators band. Execution of which is made by the undergraduate student himself not under any external company.

The final user of the device can connect her/his 433MHz sensors and actuators to the gateway, receive data and transmit commands to them respectively, via a web-based dashboard through the Internet. Moreover, the access to these resources is restricted by an RBAC[3] system along with the own platform's users she/he is able to create.

As *the icing on the cake*, a trigger of a command based on a schedule can be created in order to perform an action on an actuator periodically, e.g. gradually raising the blinds at 8 AM. Indeed, a 2019 study found that exposure to morning sunlight results in greater alertness. Morning light exposure can also lead to better sleep, which can have a cascade effect on mood the next day [3].

Moreover, an important focus is placed on the *off-grid* component. In other words, not relying on an external service is a default by design and not an opt-in. Nevertheless, users could choose to rent a network relay, in the form of a SaaS[4], so they are able to expose their devices to the public Internet without any of the hassles and vulnerabilities of opening up their routers.

The final outcome is a very professional product prototype, that could potentially be bought in a general consumer electronics shop, of which all parts of the end-to-end development have been taken into account such as the PCB, casing or even the packaging.

[1] Unlicensed bands are of public use as they aren't acquired by anyone legally. They are free to use without prior permission from the government. Wi-Fi and Bluetooth are examples of such.
[2] Bachelor's Degrees that are carried out at the university itself without any external guidance.
[3] Role Based Access Control.
[4] Software as a Service such as Cloudflare Zero Trust.

Concepts

Although this Thesis does not dive into specific details of every notion as it builds upon already existing topics, the reader should be sort of familiarized with the following main concepts to flawlessly follow up the writing.

Internet of Things

The Internet of Things (IoT) is the concept of connecting anything to the Internet and to other connected apparatus, building up a network where all devices collect and share data about the way they are used and the environment around them [4].

433MHz band / LPD433

LPD433 (low power device 433MHz) is an ultra high (UHF) radio frequency band in which license-free communication devices are allowed to operate. The operating frequencies go from 433.050 MHz to 434.790 MHz [5].

AM / ASK

Amplitude modulation (AM) is an analog modulation technique that represents data on a continuous range of voltages $[0, Vmax]$, varying the amplitude signal [6].

On the other hand, Amplitude-shift keying (ASK) is a digital modulation technique that represents data on a discrete range of voltages $\{0, Vmax\}$ (assuming two levels), varying the amplitude signal too [7].

Gateway

A gateway, in network computing terms, is any device or node that connects separate networks by translating communications from one protocol to another [8]. In this case, it converts from a 433MHz protocol signal to Internet general's HTTP, *see below*.

HTTP / REST API

The Hypertext Transfer Protocol (HTTP) is a communication protocol that allows the transmission of information between computers through a regular format and it is generally used in the Internet network based upon TCP socket primitives [9].

REST (or RESTful) APIs are Application Programming Interfaces based on certain regular formats and rules, following a Representational State Transfer architectural style and using as protocol a HTTP request-based layer [10].

Problem definition

As previously stated, the main idea that this Thesis is based on is that:

a) Belonging to the Internet of Things network provides the underlying device with a new set of capabilities being the most important three:
   i) Location transparency
   ii) Time transparency
   iii) Access transparency

   Moreover, it opens up the possibility to further extension.

b) There are a huge portion of already existing sensors and actuators operating on the 433MHz band, that perform just single-purpose actions or send information with limited monitoring, but are important enough for people to rely and use them on a day to day basis.

That being said, the problem can be defined as *how can we bring these RF devices to the Internet*, or in other words, a RF to Internet gateway.

The task I gave myself is to design and create a solution that is secure, reliable, convenient and professional to some extent.

Stakeholders

As the project is solely done by me, and not under an external business, the stakeholders involved are narrowed down to the hypothetical final users of the product, as they will be the ones directly buying and benefiting from it.

They will take advantage of the different provided functionalities, such as being notified when a movement alarm is triggered or gradually raise her/his blinds in the morning automatically, which even has positive health effects to the user as previously exposed.

Moreover, the customers will have the final word when it comes to validating the proposed features and influence the decisions taken.

1.2. Justification

Existing solutions

As of Fall of 2022 there is not much going on in this niche, moreover, every product found falls into at least one of these drawbacks:

a) Expensive: costs more than replacing every RF device to just Internet based ones.

b) Does not use 433MHz band: using instead the 800+MHz band where Zigbee, LoRa or Z-wave devices reside.

c) Vendor lock-in: only works with the company's own sensors and actuators and can only be managed through their proprietary application.

d) Closed source: functionalities cannot be extended legally without permission of the company.

e) Constant company or third party dependency: you don't have ownership or control of your data. Additionally, if the company, or other third party dependency, stops giving support to the gateway, you cannot use it anymore.

Although a market share study cannot be completed because of the lack of data, we can more or less correlate this missing information using a similar research based on their popularity.

In the following *Figure 2* there are displayed the yearly searches (in percentage) about 4 businesses that actually produce 433MHz gateways, in contrast to the searches about 433MHz bridges in the last year. Notice how the word *bridge* is intentionally used instead of *gateway* because, although meaning the same in this context, the former was more searched.

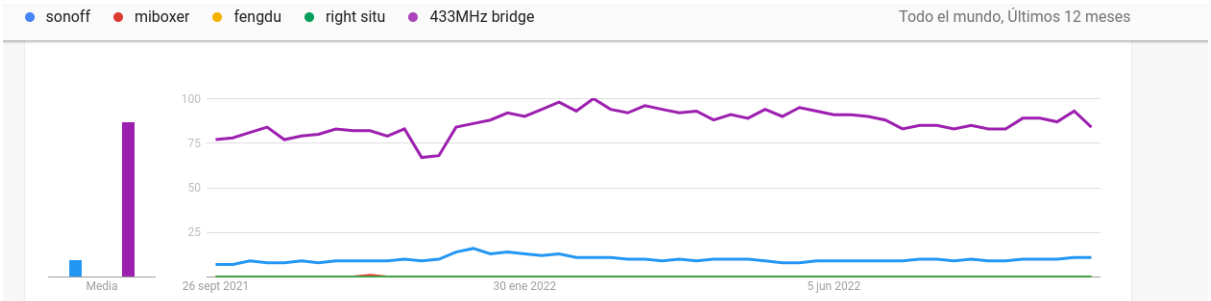


Figure 2: Percentages of global searches about Sonoff, Miboxer, Fengdu, Right Situ and 433MHz bridges on Google. *Source: Google Trends* [11].

Clearly, Sonoff dominates the scene, although it still has a long way to take over all the searches in this topic.

Value proposal

The project proposed in this Thesis tackles the previous mentioned drawbacks, and thus justifying why build it from scratch:

a) Affordable: tries to be the most inexpensive solution while maintaining all the necessary functionalities.

b) It is specifically designed for the 433MHz band.

c) Vendor agnostic: a generic abstraction interfaces with the diverse spectrum of sensors and actuators.

d) Open source: all the code and design files are publicly available to anyone wanting to take a look or extending the device furtherly.

e) Standalone: the user has the full ownership and control of its own data. Moreover, the device does not rely on any external parties to fully function.

1.3. Scope

The aim of this Thesis project is not to just design or build a bare-simple hardware prototype, but rather taking a step above, contemplating additional aspects of a retail electronic device such as manufacturing the PCB, the casing and even the packaging. All of this will end up creating a very professional product prototype that the hypothetical end user could potentially buy in a general consumer electronics shop. Notice that, it is not the final goal to make the product mass-manufacturable after the Thesis is completed.

Moreover, every aspect, aforementioned in the following *objectives* section, is curated up to a certain degree, limited by factors such as:

a) Complexity: if a short path can be taken, such as selecting already all-in-one hardware elements, and it does not heavily affect the proposed objectives, such as designing a dedicated PCB, it will be taken.

b) Economic budget: the *real*[5] expending limit for the whole development process and the manufacturing of a few samples is limited to 500 €.

c) Time deadline: the product must be finished before the Thesis presentation, that is January 27th of 2023.

Additionally, the following aspects, which reside far away from the scope of the Major, are disregarded:

a) User manuals or documentations.

b) Electrical certifications such as FCC, CE or RoHS.

c) Fine-grained compliance or other regulatory requirements (although, as required by the rubrics they will be identified in the *laws and regulations* section).

Requirements

Each product requirement, that is not already intrinsically needed to fulfill the idea, always tackles at least one of these three main pillars:

a) Security.

b) Reliability.

c) Conveniency.

[5] The amount proposed in the *budget* section is much larger that the one shown here because it seeks to get closer to a more professional budget for a hypothetical enterprise.

Functional requirements:

a) The user must be able to initially provision Internet connection to the system with her/his Wi-Fi credentials via a SoftAP[6] web-based app, and further modify them using the system's standalone web-based dashboard.

b) The user must be able to access the system's standalone web-based dashboard either through her/his private or public network.

c) The user must be able to add new 433MHz-based sensors and actuators to the system, limited to the gateway AM reception and ASK transmission technology, and further modify or delete them.

d) The user must be able to monitor generic data from a sensor in the dashboard.

e) The user must be able to send any command to an actuator through the dashboard.

f) The user must be able to create command triggers on actuators based on a certain schedule, and further modify or delete them.

g) The user must be able to create other users with roles enforcing on them access restrictions on the different sensors and actuators, and further modify or delete them.

h) The user must be able to update the system's firmware over-the-air.

Non-functional requirements:

a) Both SoftAP and dashboard interfaces must be easy to use.

b) The system's firmware must be modular, easy to maintain, modify and extend.

c) The user must be somewhat protected from any type of malicious intents performed against her/his product (IoT attacks have more than doubled over the past years [12]).

d) No external party should have access to the user's product data.

e) The user must be able to use her/his product without any significant limitations even when her/his external connection to the Internet or any external party goes down.

f) The user must be able to use her/his product in multiple different devices at any given point of time or location.

[6] Software enabled access point.

Objectives

Objectives are tightly correlated with the requirements in order to fulfill them. Every aspect of a retail electronic device has its own objective and a set of sub-objectives to complete.

Firmware:

a) Be able to encode and transmit commands to an actuator given several signal samples.

b) Be able to receive and decode data from a sensor asynchronously given several signal samples.

c) Be able to receive and respond to HTTP requests asynchronously to trigger commands on an actuator or to return sensor data.

d) Be able to initially provision, and persist, the Wi-Fi credentials through a SoftAP connection with a web app.

e) Be able to use persistent storage to dynamically add, retrieve, update or delete sensors and actuators information.

f) Be able to trigger commands on an actuator based on a schedule.

g) Be able to use persistent storage to dynamically add, retrieve, update or delete schedulements of triggering of commands on an actuator.

h) Be able to maintain the current real time with a maximum clock drift of 5s/month[7].

i) Be able to update the firmware over-the-air through an HTTP connection.

j) Be able to display connectivity and device status through a RGB LED.

k) Be able to use persistent storage to dynamically add, retrieve, update or delete user information.

l) Be able to use persistent storage to dynamically add, retrieve, update or delete roles enforcing user access rights to a sensor or an actuator.

m) Be able to access a built-in web-based dashboard allowing to manage sensors, actuators, users, roles, Wi-Fi credentials and OTA updates.

n) Be able to convert the HTTP server to HTTPS with a valid certificate.

[7] Based on the calculation of a warranty of less than 10 minutes clock drift within 10 years: (10*60)/(10*12).

o) Be able to convert the HTTP OTA to HTTPS with a valid certificate.

p) Be able to enforce firmware integrity on every boot.

q) Be able to protect firmware code and data by encrypting it.

r) Be able to convert the HTTP SoftAP provisioning to HTTPS with a valid certificate.

Schematic:

a) Select the most suitable hardware components.

b) Schematic design without any ERC[8] errors.

PCB:

a) PCB design without any DRC[9] errors.

b) PCB 3D preview without any missing hardware component.

c) Manufactured PCB sample without any electrical flaws.

Casing:

a) Casing design.

b) Casing 3D preview.

c) Manufactured case sample without any measurement or mechanical errors.

Packaging:

a) Packaging design.

b) Packaging 3D preview.

c) Manufactured package sample without any styling errors.

[8] Electrical Rule Check.
[9] Design Rule Check.

Potential obstacles and risks

Every project comes with several issues you may encounter and have to surpass in order to reach the objectives, this one is no exception:

a) Not be able to decode sensor data or actuator commands signals reliably.

b) Not be able to encode actuator commands as a valid signal reliably.

c) Not having enough resources (CPU/RAM/ROM) available in the selected microprocessor.

d) Degradation of the persistent storage (FLASH).

e) Software libraries limitations.

f) Not intuitive interfaces with bad user experience.

g) Software bugs.

h) Electrical flaws, bad PCB design or even bad physical connections.

i) Hardware components unavailability.

j) Casing and packaging measurement errors.

k) Time deadline.

l) Economic limit.

m) Inexperience, specially in embedded security (encryption).

1.4. Methodology and rigor

Time is a limited resource, having a proper working methodology is key to the development of this Thesis. Moreover, organizing the effort helps accomplish the objectives fulfilling the requirements to be met and thus finalizing the writing of the Thesis.

That's why, objectives will be broken down to a set of tasks (the sub-objectives) and each of them will be assigned a temporal estimation. These tasks will be put together into a *Kanban* framework following the *Agile* methodology.

The exact plan and timeline will be defined in the following *project planning* section. However, it must be noticed that an important task spanning the whole development of the Thesis, with the form of a weekly meeting with the Thesis director, will be held in order to discuss the status of the project, check the tasks to be accomplished during the following weeks and to ensure eliminating any blockers I may have bumped into.

By the end of the Thesis the methodology did not change, and I can ensure that the Agile Kanban methodology was the right choice for this type of project, where a single person is working by herself/himself, there is not too much time and the future is somewhat unpredictable. Specifically, it allowed me not to be tied to sprints with fixed tasks and gave me the possibility to suddenly change the planning all together as seen in the *project planning changes* section.

Validation strategy

A good validation strategy has to be simple yet effective and it has to catch the problems as soon as possible in order not to form a *snowball* with them. That’s why, every task will have its corresponding testing portion assigned in order to validate the correct development of the project.

For example, the written outcome will be periodically reviewed by the Thesis director where its feedback will be used for both validation and improvement.

On the practical side, unitary tests will be performed on each component and communication separately, followed by several end to end tests by myself and other external people to avoid bias and get feedback.

Furthermore, automatic tests will be performed by some of the following mentioned tools such as electrical schematic and PCB design rules checks or frequency band analysis.

Finally, the planning itself will be validated by checking the produced Gantt chart in the following *planning* section after each task is completed to see whether it was in time or not.

Developing, designing, analytical and monitoring tools

a) Google Meets: remote meetings with the Thesis director.

b) YouTrack: agile management and Gantt charts.

c) Google Diagrams: sketches and diagrams.

d) Visual Paradigm: flowcharts.

e) Git / GitHub: version control and code hosting.

f) VSCode: integrated development environment.

g) KiCad: schematic and PCB design.

h) Fusion360: casing and packaging design.

i) Ultimaker Cura: casing G-code[10] slicing.

j) Figma: user interface, user experience and graphic design.

k) Universal Radio Hacker: radio frequency signal analysis.

l) Gqrx: radio frequency signal analysis.

[10] Instruction language used by computer numerical control (CNC) machines such as a 3D printer.

2. Project Planning

2.1. Task list

As the project is solely done by me, and not under an external company, the daily dedication to the Thesis averages to approximately 6 hours. Obviously, this is an indicative number, there will be deviation as some days the dedication will vary depending on several factors.

Officially, the Thesis work begins with GEP on September 19th of 2022, and ends with the defense of the Thesis on January 27th of 2023. That breaks down to a total of 18 weeks and 4 days, of which 37 days correspond to weekend off time.

Therefore, the Thesis time budget sums up to 522 hours to be distributed to a total of 37 tasks.

Almost every task shown in this section refers directly to a sub-objective mentioned before in the *scope* segment. Tasks are gathered in groups alluding to the main objectives of the Thesis and are sorted by priority, impact and context switching weight. Each task has a temporal estimation and can precede or succeed another task.

Moreover, the list is summarized later on, in the form of a table and a Gantt chart.

Resources

Although every task described below already shows the material resources needed, the human resources are always present throughout the Thesis development, and they consist of 3 main individuals:

a) Undergraduate student: the actual person actively working on the Thesis.

b) Thesis director: the person to seek help from about every aspect of the Thesis document and even some technical details.

c) GEP tutor: the person in charge of correcting and guiding the project management part of the Thesis.

Additionally, global tooling such as the educational platform, Atenea; the version control system and code repository, Git and GitHub; the development environment, VSCode; and the chart designing tools, Google Diagrams and Visual Paradigm, are obviated to keep the list more concise.

Management tasks (MGT)

MGT-1 | Context and Scope *Critical Priority* | Estimation: 18 hours.

Definition and contextualization of the problem the Thesis tries to tackle on and its relevance, giving an overview of the topics the project will work on. Indication of the main objectives and the expectations with their scope.

Resources: A computer with internet connection (CIC).
Precedences: None.

MGT-2 | Planning *Major Priority* | Estimation: 18 hours.

Planning of the entire execution of the Thesis, providing a description of the phases, tasks and resources and the distribution of the work throughout the development.

Resources: A computer with internet connection and YouTrack (CYT).
Precedences: MGT-1.

MGT-3 | Budget *Major Priority* | Estimation: 7 hours.

Analysis and estimation of the cost of both human and material resources of the Thesis project, as well as the economic cost of possible unforeseen events.

Resources: A computer with internet connection.
Precedences: MGT-1.

MGT-4 | Sustainability *Normal Priority* | Estimation: 7 hours.

Analysis and elaboration of a sustainability report of the Thesis project, from environmental issues to social and economic ones.

Resources: A computer with internet connection.
Precedences: MGT-1.

MGT-5 | Integration *Normal Priority* | Estimation: 4 hours.

Bring together all the previous management tasks into an initial Thesis document, correcting the parts that were mistaken and taking into account the feedback given by the GEP tutor.

Resources: A computer with internet connection.
Precedences: MGT-1, MGT-2, MGT-3 and MGT-4.

MGT-6 | Thesis Document *Critical Priority* | Estimation: 83 hours.

A final document that is capable of explaining in a very concrete way everything related to the Thesis project and the work that has been done. The previous initial document is extended with every bit of information and documentation relevant to the project during the development of the Thesis.

Resources: A computer with internet connection.
Precedences: MGT-5 and all the following tasks in this list but one by one, that is, after completing a task its documentation part can be done.

MGT-7 | Thesis Presentation *Critical Priority* | Estimation: 36 hours.

An oral presentation with slides that allows to greatly defend the Thesis itself against a tribunal.

Resources: A computer with internet connection.
Precedences: MGT-6.

MGT-8 | Meetings *Minor Priority* | Estimation: 24 hours.

Includes every weekly meeting with the Thesis director and spans the whole development of the Thesis. They will allow me to constantly check the status of the project and its temporal deviation, as well as ensuring eliminating any blockers I may bump into.

Resources: A computer with internet connection and Google Meets (CGM).
Precedences: None or all the tasks in this list but one by one, that is, after completing a task a group of questions can be gathered and asked all at once.

Firmware tasks (FWE)

FWE-1 | Encode and transmit RF actuator commands *Major Priority* | Estimation: 30 hours.

Encode and transmit commands to an actuator given several signal samples.

Resources: The RF transmitter module and microcontroller (TXM).
Precedences: SMC-1.

FWE-2 | Receive and decode RF sensor data *Major Priority* | Estimation: 30 hours.

Receive and decode data from a sensor asynchronously given several signal samples.

Resources: The RF receiver module and microcontroller (RXM).
Precedences: SMC-1.

FWE-3 | HTTP server *Major Priority* | Estimation: 18 hours.

Receive and respond to HTTP requests asynchronously to trigger commands on an actuator or to return sensor data.

Resources: The microcontroller (MIC).
Precedences: SMC-1.

FWE-4 | HTTP SoftAP provisioning *Normal Priority* | Estimation: 15 hours.

Initially provision, and persist, the Wi-Fi credentials through a SoftAP connection.

Resources: The microcontroller.
Precedences: FWE-3.

FWE-5 | Sensor and actuator CRUD[11] *Normal Priority* | Estimation: 12 hours.

Use persistent storage to dynamically add, retrieve, update or delete sensors and actuators information.

Resources: The microcontroller.
Precedences: FWE-1, FWE-2 and FWE-3.

FWE-6 | Trigger scheduler *Major Priority* | Estimation: 12 hours.

Trigger commands on an actuator based on a schedule.

Resources: The microcontroller.
Precedences: SMC-1.

FWE-7 | Trigger CRUD *Normal Priority* | Estimation: 6 hours.

Use persistent storage to dynamically add, retrieve, update or delete schedulements of triggering of commands on an actuator.

Resources: The microcontroller.
Precedences: FWE-5 and FWE-6.

FWE-8 | NTP + RTC time sync *Minor Priority* | Estimation: 12 hours.

Maintain the current real time with a maximum clock drift of 5s/month.

Resources: The RTC module and the microcontroller (RTM).
Precedences: SMC-1.

[11] Create, Read, Update and Delete.

FWE-9 | HTTP OTA updates *Minor Priority* | Estimation: 18 hours.

Update the firmware over-the-air through an HTTP connection.

Resources: The microcontroller.
Precedences: FWE-3.

FWE-10 | Status LED *Normal Priority* | Estimation: 4 hours.

Display connectivity and device status through a RGB LED.

Resources: The RGB LED and the microcontroller (LDM).
Precedences: FWE-4 and SMC-1.

FWE-11 | User CRUD *Normal Priority* | Estimation: 6 hours.

Use persistent storage to dynamically add, retrieve, update or delete user information.

Resources: The microcontroller.
Precedences: FWE-5.

FWE-12 | Role based access control *Normal Priority* | Estimation: 10 hours.

Use persistent storage to dynamically add, retrieve, update or delete roles enforcing user access rights to a sensor or an actuator.

Resources: The microcontroller.
Precedences: FWE-5 and FWE-11.

FWE-13 | Standalone dashboard *Normal Priority* | Estimation: 30 hours.

Built-in web-based dashboard that allows to manage sensors, actuators, users, roles, Wi-Fi credentials and OTA updates.

Resources: The microcontroller and Figma (FGM).
Precedences: FWE-3, FWE-4, FWE-5, FWE-7, FWE-9, FWE-11 and FWE-12.

FWE-14 | HTTPS server *Minor Priority* | Estimation: 4 hours.

Convert the HTTP server to HTTPS with a valid certificate.

Resources: The microcontroller.
Precedences: FWE-3.

FWE-15 | HTTPS OTA updates *Minor Priority* | Estimation: 4 hours.

Convert the HTTP OTA to HTTPS with a valid certificate.

Resources: The microcontroller.
Precedences: FWE-9.

FWE-16 | Secure boot *Minor Priority* | Estimation: 4 hours.

Enforce firmware integrity on every boot.

Resources: The microcontroller.
Precedences: SMC-1.

FWE-17 | Flash encryption *Minor Priority* | Estimation: 4 hours.

Protect firmware code and data by encrypting it.

Resources: The microcontroller.
Precedences: SMC-1.

FWE-18 | HTTPS SoftAP provisioning *Minor Priority* | Estimation: 4 hours.

Convert the HTTP SoftAP provisioning to HTTPS with a valid certificate.

Resources: The microcontroller.
Precedences: FWE-4.

Schematic tasks (SMC)

SMC-1 | Component selection *Critical Priority* | Estimation: 12 hours.

Select the most suitable hardware components. Specifically, it must be chosen the central microcontroller, the RF transmitter and receiver modules, the RTC module, the RGB LED and all the other minor parts that are necessary in order to build a correct design, resistors, capacitors, connectors…

Resources: A computer with internet connection.
Precedences: MGT-1.

SMC-2 | Design *Normal Priority* | Estimation: 12 hours.

Schematic design without any ERC errors.

Resources: All the hardware components and Kicad (HWK).
Precedences: SMC-1, FWE-1, FWE-2, FWE-3, FWE-6, FWE-8 and FWE-10.

PCB tasks (PCB)

PCB-1 | Design *Normal Priority* | Estimation: 18 hours.

PCB design without any DRC errors.

Resources: All the hardware components and Kicad.
Precedences: SMC-2.

PCB-2 | 3D preview *Minor Priority* | Estimation: 12 hours.

PCB 3D preview without any missing hardware component.

Resources: All the hardware components, Kicad and Fusion360 (HKF).
Precedences: PCB-1.

PCB-3 | Manufacturing *Normal Priority* | Estimation: 6 hours.

Manufactured PCB sample without any electrical flaws.

Resources: All the hardware components, Kicad and Aisler (HKA).
Precedences: PCB-1.

Casing tasks (CSG)

CSG-1 | Design *Minor Priority* | Estimation: 12 hours.

Casing design.

Resources: All the hardware components and Fusion360 (HWF).
Precedences: PCB-2.

CSG-2 | 3D preview *Minor Priority* | Estimation: 6 hours.

Casing 3D preview.

Resources: All the hardware components and Fusion360.
Precedences: CSG-1.

CSG-3 | Manufacturing *Minor Priority* | Estimation: 6 hours.

Manufactured case sample without any measurement or mechanical errors.

Resources: The PCB assembled, Cura and a 3D printer (P3D).
Precedences: CSG-2.

Packaging tasks (PKG)

PKG-1 | Design *Minor Priority* | Estimation: 6 hours.

Packaging design.

Resources: All the hardware components, casing, Fusion360 and Figma (HFF).
Precedences: CSG-2.

PKG-2 | 3D preview *Minor Priority* | Estimation: 6 hours.

Packaging 3D preview.

Resources: All the hardware components, casing and Fusion360 (HCF).
Precedences: PKG-1.

PKG-3 | Manufacturing *Minor Priority* | Estimation: 6 hours.

Manufactured package sample without any styling errors.

Resources: The PCB and its casing assembled together and Packly (PCP).
Precedences: PKG-2.

Summary

The following *Table 1* summarizes all tasks to be done in the Thesis project. Each one has its own identifier, name, temporal estimation in hours, priority, its dependencies and resources needed. The previous section contains a more detailed description of each task.

ID	NAME	HOURS	PRIO	DEPENDENCIES	RES
MGT	**Management**	**197**			
MGT-1	Context and Scope	18	CRI	None	CIC
MGT-2	Planning	18	MAJ	MGT-1	CYT
MGT-3	Budget	7	MAJ	MGT-1	CIC
MGT-4	Sustainability	7	NOR	MGT-1	CIC
MGT-5	Integration	4	NOR	MGT-1, MGT-2, MGT-3, MGT-4	CIC
MGT-6	Thesis Document	83	CRI	MGT-5 and following one by one	CIC
MGT-7	Thesis Presentation	36	CRI	MGT-6	CIC
MGT-8	Meetings	24	MIN	None or all one by one	CGM
FWE	**Firmware**	**223**			
FWE-1	Encode and transmit RF	30	MAJ	SMC-1	TXM
FWE-2	Receive and decode RF	30	MAJ	SMC-1	RXM
FWE-3	HTTP server	18	MAJ	SMC-1	MIC
FWE-4	HTTP SoftAP provision	15	NOR	FWE-3	MIC
FWE-5	Sensor & actuator CRUD	12	NOR	FWE-1, FWE-2, FWE-3	MIC
FWE-6	Trigger scheduler	12	MAJ	SMC-1	MIC
FWE-7	Trigger CRUD	6	NOR	FWE-5, FWE-6	MIC
FWE-8	NTP + RTC time sync	12	MIN	SMC-1	RTM
FWE-9	HTTP OTA updates	18	MIN	FWE-3	MIC
FWE-10	Status LED	4	NOR	FWE-4, SMC-1	LDM
FWE-11	User CRUD	6	NOR	FWE-5	MIC
FWE-12	Role based access control	10	NOR	FWE-5, FWE-11	MIC
FWE-13	Standalone dashboard	30	NOR	FWE-3 … FWE-7, FWE-9, FWE-11, FWE-12	FGM
FWE-14	HTTPS server	4	MIN	FWE-3	MIC
FWE-15	HTTPS OTA updates	4	MIN	FWE-9	MIC
FWE-16	Secure boot	4	MIN	SMC-1	MIC
FWE-17	Flash encryption	4	MIN	SMC-1	MIC
FWE-18	HTTPS SoftAP provision	4	MIN	FWE-4	MIC
SMC	**Schematic**	**24**			
SMC-1	Component selection	12	CRI	MGT-1	CIC
SMC-2	Design	12	NOR	SMC-1, FWE-1 … FWE-6, FWE-8, FWE-10	HWK
PCB	**PCB**	**36**			
PCB-1	Design	18	NOR	SMC-2	HWK
PCB-2	3D preview	12	MIN	PCB-1	HKF
PCB-3	Manufacturing	6	NOR	PCB-1	HKA
CSG	**Casing**	**24**			
CSG-1	Design	12	MIN	PCB-2	HWF
CSG-2	3D preview	6	MIN	CSG-1	HWF
CSG-3	Manufacturing	6	MIN	CSG-2	P3D
PKG	**Packaging**	**18**			
PKG-1	Design	6	MIN	CSG-2	HFF
PKG-2	3D preview	6	MIN	PKG-1	HCF
PKG-3	Manufacturing	6	MIN	PKG-2	PCP

Table 1: Summary of the tasks, where CRI priority means CRITICAL; MAJ: MAJOR; NOR: NORMAL; and MIN: MINOR. *Source: Compiled by the author*.

2.2. Gantt chart

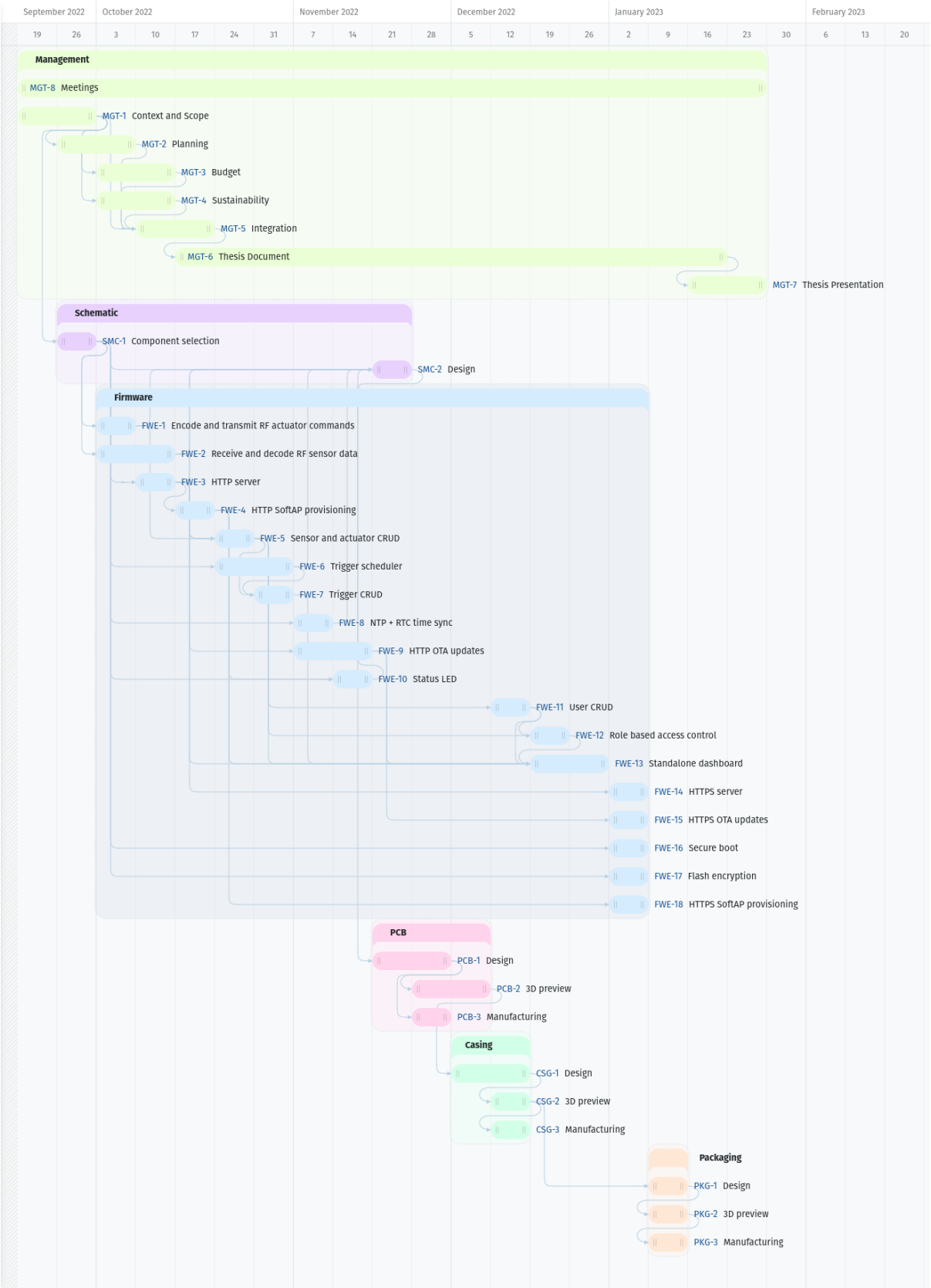


Figure 3: Gantt chart of the tasks. *Source: Compiled by the author*.

The previous *Figure 3* shows the Gantt chart of all the tasks of the Thesis project. Management tasks are in light green, schematic tasks in purple, firmware tasks in blue, PCB tasks in pink, casing tasks in dark green and packaging tasks in orange.

The diagram starts on September 19th 2022 and ends on January 27th 2023 with the Thesis defense.

It is necessary to specify that:

a) YouTrack does not allow recurring tasks, so tasks such as meetings and the Thesis documentation span through all the diagram instead of intermittently appear.

b) YouTrack has compressed the chart to time periods of weeks and it could seem as if most of the tasks are concurrent, but in reality are sequential but on the same week.

2.3. Risk management

The potential obstacles and risks were already mentioned in their own section. Nonetheless, good planning requires taking into account these threats and its countermeasures in order to estimate their impact and be able to reach the proposed deadline.

Moreover, if the above is not sufficient, practical tasks are planned to be finished 2 weeks before the day of the Thesis presentation.

Firmware risks

These risks include general software obstacles such as bugs, counterintuitive interfaces or libraries limitations and more specific issues such as not being able to decode or encode sensor data and actuator commands signals reliably.

<u>Impact</u>

For the first ones it is hard to estimate their impact until I encounter them, because they could be minor problems that don't even have to be solved. Also, their solution could take 5 minutes of debugging or even hours.

The second ones on the other hand, are issues that would be critical for the completion of the Thesis. They are essentially blockers that have to be taken into special consideration.

<u>Countermeasure</u>

General errors are countered by having a testing portion in every task, not only unitary but also UI testing. 18 FWE tasks, 1.5 hours testing each, +27 hours.

For the second ones, data and command signals will be recorded with a special program and hardcoded into the code as an extreme solution. +3 hours.

Hardware risks

These risks consist of *physical* problems such as electrical flaws, measurement errors or even bad physical connections; and *logistic* issues caused by hardware components unavailability.

Impact

The impact from physical problems could vary from momentary: bad connections can be traced down and repaired; to lasting: the PCB has a major flaw and has to be remanufactured.

About the logistical problem, it actually has little impact if taken into account with sufficient time, but it also depends on the importance of the missing components.

Countermeasure

Physical problems are countered by having a testing portion in every task. 9 PCB, CSG and PKG tasks, 1.5 hours testing and fixing each, +14 hours.

Logistical problems are solved by preselecting alternatives for every required component. +10 hours of searching and buying.

Limiting risks

These risks include both the time deadline and the economic limit.

Impact

The budget is set pretty high. Additionally, the hardware components will be selected accordingly, so this risk is very improbable, and thus the impact is reduced.

On the other hand, the time deadline is a hard limit and the project is pretty ambitious. Finishing the Thesis halfway would have an enormous impact on the final result.

Countermeasure

Tasks are already prioritized, so, if the schedule starts to fail, the lower priority tasks, starting with the security ones, will be moved to the *future work* section. No hours change.

Personal risks

It basically consists of the inexperience in the field, specially in embedded security.

Impact

Security is the last of the main objectives of this Thesis, lies just next to the limit of the scope and the device can function flawlessly without the security bits, so the impact is reduced.

Countermeasure

If the task's complexity becomes a blocker they will be moved to the *future work* section. Maximum +10 hours for learning.

2.4. Changes

In regard to the aforementioned proposed project planning, there was a major change on the task schedule just at the end of the project management phase. The content, estimation and the number of them, however, did not suffer any change whatsoever. All tasks fitted with clearance the objectives and covered the proposed requirements with clearance.

About the changes made, there were two things that I did not take into account when planning the tasks: third-party dependencies on the PCB and packaging manufacturers. I did take into account the other main external dependencies such as the hardware component availability and provisioning, by selecting and buying the components parallelly with the project management tasks, right at the start of the Thesis; or the casing manufacturing as I already owned a 3D printer.

The unpredictable and possibly long lead times along with the fact that not having the PCB manufactured in time was a giant blocker for the completion of the product prototype and thus the Thesis, made me realize that all hardware-related tasks: SMC, PCB, CSG and PKG, should be prioritized instead of advancing furtherly on the firmware, as in that realm there is way more flexibility and control on when it can be done.

Obviously, before going into the schematic and PCB design, there were critical tasks that had to be finished and validated first, selecting the components, SMC-1, and ensuring correct firmware interaction with all the external hardware modules such as the RF transmitter, FWE-1; the RF receiver, FWE-2; the RTC, FWE-8; and the status LED, FWE-10.

Revisited gantt chart

With all of the above in mind, the following *Figure 4* shows the revisited Gantt chart of all the tasks of the Thesis project that better aligns on the reality of what happened. Remember that YouTrack has compressed the chart to time periods of weeks and it could seem as if most of the tasks are concurrent, but in reality are sequential but on the same week.

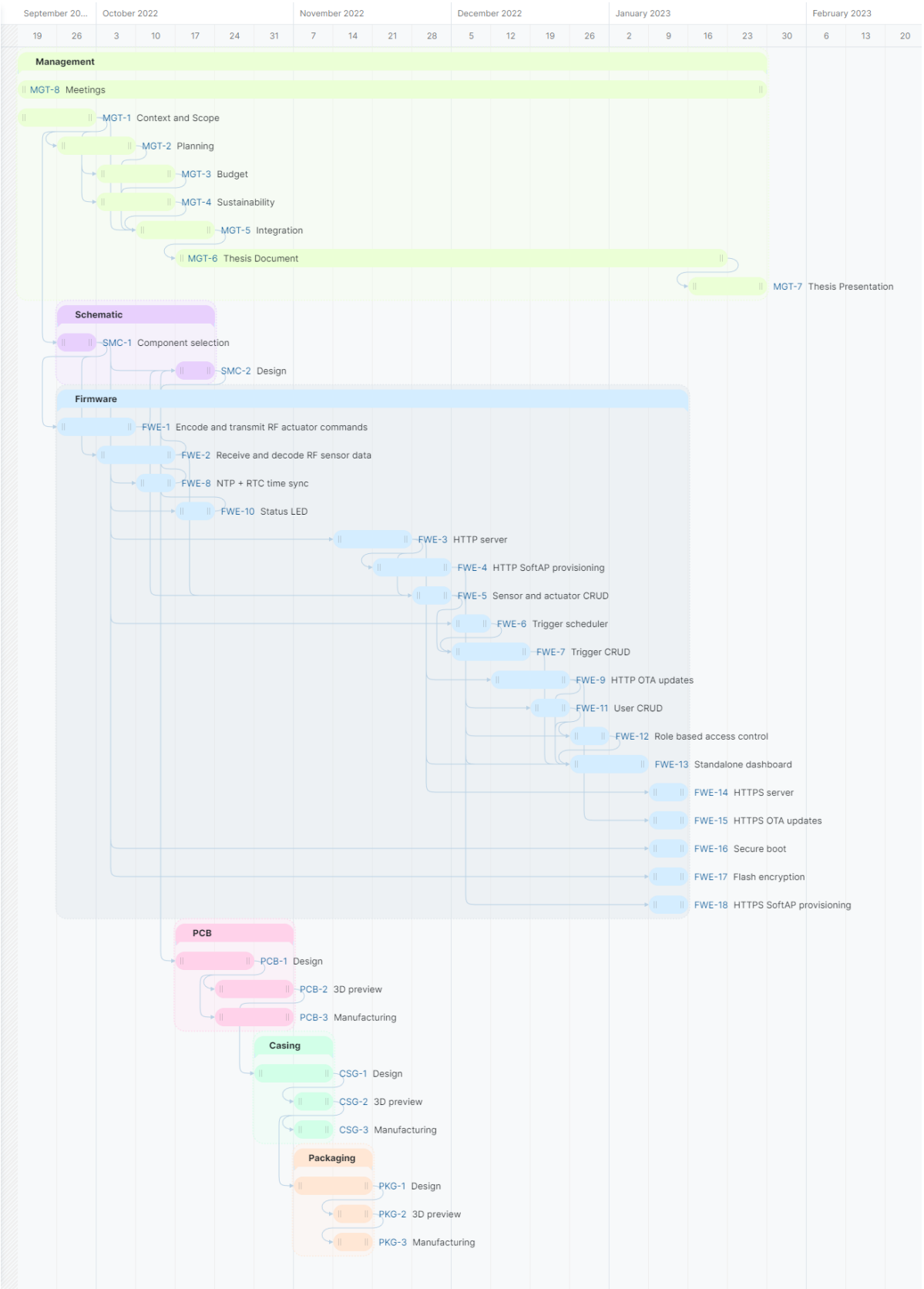


Figure 4: Revisited Gantt chart of the tasks. *Source: Compiled by the author*.

3. Budget

3.1. Cost identification and estimation

The project that is presented by this Thesis is a very *physical* (practical) one, spanning from the computer screen to the real world. The goal is to develop from start to finish a potential electronic consumer product, and therefore, it carries a series of direct, indirect and even hidden costs, that have to be thoroughly thought of and where even their complexity varies depending on factors such as manufacturing or economic scalability.

That's why, every cost has to be correctly identified, and estimated, based on the tasks previously proposed in the *project planning* section. Additionally, potential incidents must be also contemplated, thus a contingency plan must be elaborated.

This ends at aggregating all the costs into a final budget, presented later on this section.

However, the real world presents surprises and complexities, which can make subtle modifications to the proposed budget. These deviations must be quickly recognized and controlled through a robust mechanism.

Staff costs

In a professional scenario several different profiles would be managing and executing the project. However, it will be simplified down to the most essential four and the student will be playing all fourth roles himself:

a) Project manager: the person in charge of organizing, controlling and leading the project (MGT tasks). At the meetings the Thesis director will take over the role.

b) Computer engineer: the hardware specialist in charge of selecting the components, developing the firmware (FWE tasks), designing the electrical schematic (SMC tasks) and the printed circuit board (PCB tasks).

c) Industrial engineer: the mechanical and producer expert in charge of designing and manufacturing the product enclosure (CSG tasks) and package (PKG tasks).

d) Web developer: the programmer master at web frontend technologies in charge of designing and building the built-in dashboard (FWE-13 task).

The following *Table 2* shows the estimated gross salary in Spain for the previous job profiles as of Fall of 2022 (assuming a normal 40 hours workweek), along with the real final employer cost including the contributions to the Social Security system and other withheld taxes, summing up to around 1.35x increase.

ID	NAME	GROSS SALARY		EMPLOYER COST
		ANNUAL	HOURLY	
PM	Project Manager	46,200	24.00	32.40
CE	Computer Engineer	38,400	20.00	27.00
WD	Web Developer	31,600	16.50	22.27
IE	Industrial Engineer	37,900	19.75	26.66

Table 2: Average gross salary and final employer cost in euros in Spain, for project managers, computer engineers, web developers and industrial engineers. *Source: Jobted* [13].

The next *Table 3* depicts the amount of hours that each profile must dedicate to each of the project tasks along with its final cost using the previous shown hourly employer cost. The hours concur with the ones shown in the *project planning* section, although the meeting time spans through all four workers as they would be present in all of them.

ID	NAME	HOURS				COST
		PM	CE	WD	IE	
MGT	**Management**	**197**	**24**	**24**	**24**	**8,205.12**
MGT-1	Context and Scope	18				583.20
MGT-2	Planning	18				583.20
MGT-3	Budget	7				226.80
MGT-4	Sustainability	7				226.80
MGT-5	Integration	4				129.60
MGT-6	Thesis Document	83				2,689.20
MGT-7	Thesis Presentation	36				1,166.40
MGT-8	Meetings	24	24	24	24	2,599.92
FWE	**Firmware**	**0**	**193**	**30**	**0**	**5,879.10**
FWE-1	Encode and transmit RF		30			810.00
FWE-2	Receive and decode RF		30			810.00
FWE-3	HTTP server		18			486.00
FWE-4	HTTP SoftAP provision		15			405.00
FWE-5	Sensor & actuator CRUD		12			324.00
FWE-6	Trigger scheduler		12			324.00
FWE-7	Trigger CRUD		6			162.00
FWE-8	NTP + RTC time sync		12			324.00
FWE-9	HTTP OTA updates		18			486.00
FWE-10	Status LED		4			108.00
FWE-11	User CRUD		6			162.00
FWE-12	Role based access control		10			270.00
FWE-13	Standalone dashboard			30		668.10
FWE-14	HTTPS server		4			108.00
FWE-15	HTTPS OTA updates		4			108.00
FWE-16	Secure boot		4			108.00
FWE-17	Flash encryption		4			108.00
FWE-18	HTTPS SoftAP provision		4			108.00
SMC	**Schematic**	**0**	**24**	**0**	**0**	**648.00**
SMC-1	Component selection		12			324.00
SMC-2	Design		12			324.00
PCB	**PCB**	**0**	**36**	**0**	**0**	**972.00**
PCB-1	Design		18			486.00
PCB-2	3D preview		12			324.00
PCB-3	Manufacturing		6			162.00

CSG	Casing	0	0	0	24	639.84
CSG-1	Design				12	319.92
CSG-2	3D preview				6	159.96
CSG-3	Manufacturing				6	159.96
PKG	**Packaging**	**0**	**0**	**0**	**18**	**479.88**
PKG-1	Design				6	159.96
PKG-2	3D preview				6	159.96
PKG-3	Manufacturing				6	159.96
TOTAL		**197**	**276**	**54**	**66**	**16,823.94**

Table 3: Cost in euros and hours dedicated to project tasks by profile.
Source: Compiled by the author.

Material costs

Developing an electronic product implies several hardware components and manufacturing services, this project is not an exception. In the following *Table 4* are shown the different cost, as of Fall of 2022, of the parts that form the final product, all prices are taken out of the electronics provider *DigiKey* [14]. More details of each chosen hardware element and full BOM[12] can be found in the *hardware components* section.

Just to show the difference in cost between a wholesale or retail component order, the unitary price of the same item in a mass purchase is also included (below the *x1000* column). As seen, it cuts the total component cost by half thus the manufacturing and economic scalability is illustrated.

The services used to produce both the PCB and packaging are also included as material costs, as they are as well costs per unit of product manufactured. These are:

a) Aisler: German PCB manufacturer. Prices taken out of the provider *itself* [15]. More details on the choice of manufacturer in the *PCB* section.

b) Packly: Italian packaging manufacturer. Prices taken out of the provider *itself* [16]. More details on the choice of manufacturer in the *packaging* section.

Bulk and single costs are also included here as the differences in prices heavily depicts the manufacturing and economic scalability once again.

Moreover, several tools will be needed throughout the development of the Thesis as well:

a) ESP32-S3-DEVKITC-1-N32R8: the development kit for the selected microcontroller. It allows to develop all the firmware in a simpler and faster way.

b) ACER ASPIRE 3: laptop with sufficient resources. It allows to do the management tasks, the firmware and all required designs.

[12] Bill of Materials.

c) AN8009: digital multimeter. It allows to check for electric flaws in the final PCB and also to debug issues with the hardware components.

d) YIHUA 926 III: soldering iron. It allows to solder the hardware components into the manufactured PCB.

e) ENDER 3: 3D printer. It allows to manufacture the product enclosure (casing).

Tool pricing is taken out of the e-marketplace *Amazon* [17].

Mostly all items will be amortized in roughly 10 years, that is 120 months, except for the laptop, soldering iron and 3D printer, which are estimated to be half. For this budget it only matters the amortization in the 18 weeks of the Thesis duration, that is 4.5 months.

ID	NAME	UNITS	COST		LIFE TIME	AMORTIZATION	
			x1	x1000		MONTHLY	THESIS
COMPONENTS			**35.95**	23.39			
E3W	ESP32-S3-WROOM-2-N32R8V	1	7.34	7.34	120	0.06	0.27
QR4	QAM-RX10-433	1	4.38	2.46	120	0.03	0.13
QT4	QAM-TX2-433	1	5.51	2.63	120	0.04	0.18
W31	W3127	2	1.90	0.88	120	0.03	0.13
D33	DS3231SN	1	4.06	2.80	120	0.03	0.13
CFK	CLX6F-FKC	1	0.34	0.26	120	0.01	0.04
BC8	BC847B	4	0.16	0.06	120	0.01	0.04
USB	USB4085	1	0.75	0.62	120	0.01	0.04
UC6	USBLC6-2SC6	1	0.65	0.29	120	0.01	0.04
LDL	LDL1117S33R	1	1.38	0.79	120	0.01	0.04
MP0	MP008293	1	0.11	0.05	120	0.01	0.04
MFM	MF-MSMF075/24	1	0.32	0.30	120	0.01	0.04
1N5	1N5819HW-7-F	1	0.53	0.10	120	0.01	0.04
KMR	KMR421G-LFS	2	0.72	0.72	120	0.01	0.04
R85	RESISTORS SMD 0805	19	0.01	0.01	120	0.01	0.04
C85	CAPACITORS SMD 0805	12	0.09	0.01	120	0.01	0.04
I85	INDUCTORS SMD 0805	4	0.86	0.50	120	0.02	0.09
SERVICES			**18.20**	3.85			
APB	Man. PCB (AISLER)	1	9.56	3.11	120	0.08	0.36
PPK	Man. PACKAGING (PACKLY)	1	8.64	0.74	120	0.07	0.31
TOOLS			**676.55**				
E3D	ESP32-S3-DEVKITC-1-N32R8V	1	17.97		120	0.15	0.67
AA3	ACER ASPIRE 3	1	349.00		60	5.81	26.14
AN8	AN8009	1	50.59		120	0.42	1.89
Y93	YIHUA 926 III	1	49.99		60	0.83	3.73
ER3	ENDER 3	1	209.00		60	3.48	15.66
TOTAL			**730.70**			**11.16**	**50.12**

Table 4: Cost in euros of project hardware components, services and tools needed along with their lifetime in months and amortization in euros. *Source: Compiled by the author*.

Generic costs

Basic resources and services such as electricity, internet, software licenses, electronic workshop and coworking space rent are needed to be able to develop the project. The workshop is needed for the PCB, casing and packaging manufacturing tasks and the coworking for the physical meetings.

However, almost half of the meetings can be done remotely, so the coworking space will only be rented for half the meeting hours.

Moreover, the software that will be used is either open source or has an extensive free tier license, so it can be neglected.

The electricity cost has been computed using the day-of-writing 0.32 €/kWh average cost from *Selectra* [18] and the consumption based on the tools used with the following formula: $P_{E3D} * T_{FWE} + P_{AA3} * T_{ALL} + P_{AN8} * T_{PCB\text{-}3} + P_{Y93} * T_{PCB\text{-}3} + P_{ER3} * T_{CSG\text{-}3}$ where P represents the power (W) of the tool and T the total time (h) of the tasks. The consumption of the development kit is minimal, and thus neglected, and the multimeter has a cell battery, the other powers are 45W, 60W and 125W respectively.

The internet plan price, 30 €/month, is taken out of *Kelisto* [19], the workshop rent, 35 €/day, from *Makerspace Barcelona* [20] and the coworking rent, 16 €/hour, from *Itnig* [21].

The following *Table 5* shows the summary of the different generic costs through the 4.5 months of the Thesis.

ID	NAME	COST
ELE	Electricity	7.87
INT	Internet	135.00
EWR	Electronic workshop rent	105.00
CWR	Coworking rent	192.00
SWE	Software	0.00
TOTAL		**439.87**

Table 5: Cost in euros of electricity, internet, space rent and software through all the project.
Source: Compiled by the author.

Incidental costs

Possible incidences have already been described in the previous *potential obstacles and risks* and *risk management* sections. Moreover, the possibility of any tool or the hardware components breaking are added.

The following *Table 6* shows these incidentals along with their full cost, probability and the final amount to add to the budget.

ID	NAME	IMPLICATION	PROB. %	COST	
				FULL	BUDGET
BDK	Broken development kit	Buy again	5	17.97	0.89
BPC	Broken computer	Buy again	1	349.00	3.49
BMU	Broken multimeter	Buy again	1	50.59	0.50
BSI	Broken soldering iron	Buy again	1	49.99	0.49
B3D	Broken 3D printer	Buy again	5	209.00	10.45
BHW	Broken components (all)	Buy again	10	42.14	4.21
UHW	Unavailable components (all)	Find and buy alternatives	33	50.00	16.50
LSW	Software libraries limitations	Find alternatives and refactor (15h)	5	405.00	20.25
BSW	Software bugs	Find and fix (15h)	20	405.00	81.00
FPB	PCB flaw	Fix and remake (10h)	15	270.00	40.50
FCS	Casing flaw	Fix and remake (8h)	10	213.28	21.32
FPK	Packaging flaw	Fix and remake (6h)	10	159.96	15.99
IEX	Inexperience	Learn (10h)	20	253.10	50.62
DEA	Cannot reach deadline	Do extra work (20h)	20	541.65	108.33
TOTAL					**374.53**

Table 6: Cost in euros and their probability of the possible incidences through all the project. *Source: Compiled by the author*.

Contingency costs

Even after taking into account the upper bounds of all the different costs and the impact on the budget of the potential incidents, the future cannot be predicted with certainty. That's why anticipating the possible situation where things don't go as planned or the costs heavily vary (electricity) a security margin of 20% must be added to the final budget.

That's a total of 3,537.69 €.

3.2. Final budget

The following *Table 7* shows the different costs aggregated by type and its total final value. On the material type is shown the amortized cost.

ID	NAME	COST
STF	Staff	16,823.94
MAT	Material	50.12
GEN	Generic	439.87
INC	Incidental	374.53
CGY	Contingency	3,537.69
TOTAL		**21,226.15**

Table 7: Cost in euros of staff, materials, generics, incidences and contingencies through all the project. *Source: Compiled by the author*.

3.3. Management control

Deviations to the presented budget will be quickly recognized and controlled through the following formula: $\sigma_{TOT} = \sigma_{STF} + \sigma_{MAT} + \sigma_{GEN} + \sigma_{INC}$ where each σ_n refers to the deviation of the previous presented budget by ID, and calculated using $\sigma_n = COST_{PAID} - COST_{EXPECTED}$.

If at the end of the project $\sigma_{TOT} \leq 0$ the proposed budged will have succeeded. Otherwise, it will result in the extra unforeseen amount paid.

This formula must be reviewed in every weekly meeting to check for accumulated deviations on any budget by ID ($\sigma_n \leq 0$ would also mean that budget ID is OK).

4. Sustainability

4.1. Self assessment

Now, more than even before, it's our task to take care of the future and take sustainability into account before starting any type of project. That's why assessing sustainable development of this Thesis in its whole three dimensions: economic, environmental and social; it's a must, and it will be done using the questions of the sustainability matrix provided on the rubric at the start and at the end of the Thesis project.

In order to evaluate sustainability self-awareness, the research project *EDINSOST2-ODS Sustainable Development Goals* questionnaire [22], backed by the Ministry of Science and Innovation, the State Research Agency and FEDER funds, has been answered.

Honestly, before completing the survey, I directly associated the concept of sustainability with only environmental preservation, reducing emissions or decreasing power consumption. However, sustainability is a very complex field, where its concerns span through the environmental issues themselves, social security and equality affairs or even the proper economical viability, among others.

The questionnaire made me realize how little I knew about sustainability and gave me examples of global initiatives, like the *2030 Agenda for Sustainable Development* [23] and its *17 goals*.

Moreover, I bumped into several sustainability indicators and metrics that will allow me to assess this Thesis project and the ones that will come after, putting my *little grain of sand*.

4.2. Economic dimension

Have you estimated the cost of undertaking the project (human and material resources)?

In the previous *budget* section all the different costs of this Thesis project have been assessed, including both human (staff) and material resources among other such as incidentals and contingencies. Moreover, in order to show the manufacturing and economic scalability of an electronic product, the wholesale unitary price for each hardware component and manufacturing service has also been included.

Additionally, in order to control the deviations of the proposed budget, a management control has been designed and described.

Have you quantified the cost (human and material resources) of undertaking the project? What decisions have you taken to reduce the cost? Have you quantified these savings?

Through the development of the Thesis I did track every cost of all hardware components and services I used. About the electricity cost, as it heavily fluctuates every day, I used as an average the price of-the-day of the aforementioned budget.

Moreover, several shortcuts have been taken in order to reduce the costs to some extent:

a) Select the best quality-price hardware components.

b) Manufacture the casing myself in order to reuse already owned resources.

c) Manufacture both the PCB and packaging inside the European territory, Germany and Italy respectively, to be exempt from import and duty taxes and reduce shipping costs.

As prices fluctuate I cannot give an absolute value for the potential savings. However, the cuts could sum up from 5% to 10% of the final unitary price of the product, 54.15 € as stated before in the *material costs* section.

However, enormous savings can further be done if we take into account the manufacturing and economic scalability of the resources. As stated before, wholesale costs of hardware components and manufacturing services decrease exponentially with the numbers of units produced. Taking into account the numbers presented in the *budget* section, the potential savings are more than 50% of the final unitary price.

Is the expected cost similar to the final cost? Have you justified any differences (lessons learnt)?

At the end of this Thesis I can now ensure that the estimated budget fitted well the reality of the situation. Obviously, all the personnel, tools I already owned, almost all generic, incidental and contingencies related costs were not a part of the real final expenses, as it was estimated for a hypothetical professional business.

Therefore, the final cost is the sum of the components and services material cost and the electricity and internet generic cost, that is a total of 197.02 €.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution economically improve existing solutions?

As stated before in the *justification* section, a market share study was carried over. In the research it was shown that, although this project builds upon pretty known topics, there are not many available solutions, and thus there is a fair amount of market space left.

Economically speaking, it is true that these solutions are not that expensive. However, although it is not the main intention of this Thesis, the pricing factor has been always taken into consideration, though obviously requiring more future work for lowering the product cost.

What cost do you estimate the project will have during its useful life? Could this cost be reduced to increase viability?

The only critical thing that the product needs is electricity, and so is the main cost that it will have during its useful life. This electricity cost can be reduced with a better electrical design that lowers power consumption. However, the proposed product is already built upon low consumption hardware parts and it is not a kind of device that requires great amounts of power anyways.

Another optional thing that the product needs to operate is access to the Internet. Although, as said it is optional as it can fully operate without it as it was one of the main objectives of this Thesis. This cost can be reduced if Internet plans become cheaper gradually.

However, both of these costs fall on the consumer end and thus, the hypothetical business does not need to take care of them.

Have you considered the cost of adaptations/updates/repairs during the useful life of the project?

On the one hand, as stated before in the *objectives* section, one of the requirements is that the user has to be able to update the device over-the-air. That means, the cost in time and money of firmware updates is minimal.

On the other hand, the ease of hardware repair does not fit inside the scope of the project, which is already pretty ambitious, as is not the final goal to mass produce it anyways. However, the simplicity of the proposed design should help anyone willing to repair her/his product with some help.

Could situations occur that are detrimental to the project's viability?

Economically speaking, the main harmful situation is having problems on the hardware components supply chain. If one single component is unavailable the product line is blocked until the component is available again or swapped.

Another critical situation could happen if another player suddenly takes over all the market for this kind of product. It could even happen that suddenly, people stop using the 433MHz devices altogether.

4.3. Environmental dimension

Have you estimated the environmental impact of undertaking the project? Have you considered how to minimize the impact, for example by reusing resources?

The proposed product is already built upon low consumption hardware parts and it is not a kind of device that requires great amounts of power anyways.

Furtherly optimizing the power consumption does not fit inside the scope of the project, which is already pretty ambitious. Nevertheless, that does not mean that it has not been taken into consideration, as possible solutions like switching off selectively different parts of the device, such as Wi-Fi or RF communications, have already been thought of for future work.

However, environmental impact can be directly estimated by measuring the average power consumption of the product in all of its states, which correlates directly with the energy drawn from the system and its consequential carbon footprint.

Have you quantified the environmental impact of undertaking the project? What measures have you taken to reduce the impact? Have you quantified this reduction?

The impact can be quantified but is a very complex task.

On the one hand, the most direct impact is the power consumption (wattage) of the finished product as a whole. That can be easily measured and it is around 0.4 W as stated later on in the *results* section. This can indeed be reduced. However, furtherly optimizing the power consumption does not fit inside the scope of the project, which is already pretty ambitious. But possible solutions like switching off selectively different parts of the device, such as Wi-Fi or RF communications, have already been thought of for future work.

On the other hand, the other hidden cost, that probably has even more environmental impact, is the carbon footprint of both producing the components that form the product and producing the product itself. Measuring this is a very complex task, as impact data is generally not publicly accessible. Moreover, it is really not a thing I can try to reduce on.

If you carried out the project again, could you use fewer resources?

I genuinely think I used the fewest resources as I possibly could and I used them thoroughly, as it is also beneficial for me. Using more expensive or high quality components, services or tools would end up eating a budget I would have to pay by myself.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution environmentally improve existing solutions?

As said before, the few players in this field have pretty similar solutions, almost identical. Because of the lack of data, no assumptions can be made, but no indications have been found either to affirm environmental issues are being addressed.

From the future work described in the last question, the proposed product will furtherly try to simplify its system down to its basics, and thus improving its environmental impact.

What resources do you estimate will be used during the useful life of the project? What will be the environmental impact of these resources?

As mentioned before in the economical dimension part, the product only really needs electricity to work, and an optional Internet connection to enhance its functionalities.

The impact that both of these resources would have really depends on the contracted services of the end users that buy the product. Although, I believe that little by little in the future, more and more renewable energy sources will be used.

Will the project enable a reduction in the use of other resources? Overall, does the use of the project improve or worsen the ecological footprint?

Directly, I think that the product worsens the ecological footprint by a little, as it is an electronic product that has to be built and constantly consumes electricity. Although, it saves the most important resource the humans have, time.

Indirectly, however, it is possible that some hypothetical end user uses this device in order to better manage her/his sensors and actuators, ending up on saving electricity or other harmful resources.

Could situations occur that could increase the project's ecological footprint?

Increasing the ecological footprint really depends on the hardware components supply chain and production line. If both of these areas get worse in the future, it will directly affect the general footprint of the presented product.

4.4. Social dimension

What do you think undertaking the project has contributed to you personally?

As a self-defined maker and a person that loves to deep dive into a problem that people actually have, understand it in its fullest, creating a solution that people will use and doing all of this while joining my two passions, software and hardware, I will not only enjoy from doing the Thesis but it will also step up my game in areas that I don't feel comfortable with, such as designing CAD models, that in the professional world are must-to-have skills.

Moreover, as odd as it seems, I feel this project has the potential to turn into a viable business.

Has undertaking this project led to meaningful reflections at the personal, professional or ethical level among the people involved?

I personally don't think that this Thesis project has a deep ethical meaning whatsoever, as it is the construction from scratch of a retail electronic consumer device prototype.

However, as stated in the previous question, as a self-defined maker and a person that loves to deep dive into a problem that people actually have, not only I have greatly enjoyed doing the Thesis but it also has stepped up my game in areas that I don't feel comfortable with.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution socially improve (quality of life) existing solutions?

The internet of things ecosystem has always been designed to improve the quality of life of people, from less pollutants industries, more efficient agriculture, more health issues awareness to a simpler way of living thanks to smarter houses.

It's in this last objective where the current solutions and my project reside. They allow to enhance the current RF devices, that people has all around their houses and are important enough for them to rely and use them on a day to day basis, simplifying the communication and managing of these devices.

Specifically the proposed project tackles this simplification, by offering a painless interface and experience, while providing more complex features, such as being able to trigger a command on an actuator periodically based on a schedule, e.g. gradually raising the blinds at 8 AM. As previously stated in the *contextualization* section this simple action leads to better sleep, which can have a cascade effect on mood the next day.

Is there a real need for the project?

From the data shown at the start of the Thesis in the *contextualization* section, we can clearly see that there exists a huge portion of RF devices that can be smartified by hooking them up to the Internet and are important enough for people to rely and use them on a day to day basis.

Moreover, this can be asserted by the global interest study carried over in the *justification* section, where the popularity is not only there but it is increasing year by year.

On the other hand, the value proposition, also stated before in the *justification* section, fulfills the needs of these people. All in all, I think there is a real need for the project's product.

Who will benefit from the use of the project? Could any group be adversely affected by the project? To what extent?

As said before in the *stakeholders* section, the hypothetical final users of the product will be the ones benefiting from it.

They will take advantage of the different provided functionalities, such as being notified when a movement alarm is triggered or gradually raise her/his blinds in the morning automatically, which even has positive health effects to the user as previously exposed.

I do not believe that any group of people will be negatively affected by the existence of this product whatsoever.

To what extent does the project solve the problem that was established initially?

As seen in the *results* section of this Thesis, I strongly believe that this project was a great success and it does solve with clearance all the proposed objectives and requirements.

The final user of the device can connect her/his 433MHz sensors and actuators to the gateway, receive data and transmit commands to them respectively, via a web-based dashboard through the Internet. Moreover, a trigger of a command based on a schedule can be created in order to perform an action on an actuator periodically.

Could situations occur in which the project adversely affects a specific population segment? Could the project create any kind of dependency that puts users in a weak position?

I do not believe that any specific population segment nor anyone in general will be adversely affected or put them in any type of discriminatory or weak position by the existence or the usage of this product. I cannot think of any harmful situation.

5. Laws and Regulations

As stated in the *scope* section, it is not the aim of this Thesis to comply with the different laws and regulations. Notice that, it is not the final goal to make the product mass-manufacturable after the Thesis is completed.

Moreover, on electronic products, compliance tends to be both time and money consuming, as the hardware, casing and packaging regulations add up having a lot more directives than its software counterpart.

However, identifying the distinct normatives related to the project and their requirements is critical to the development path of the product prototype, as it can affect the general vision and the decided global objectives.

Compliance for this Thesis can be splitted into two main areas: the software, or in this case the firmware; and the hardware, which contemplates the other four topics: schematic, pcb, casing and packaging. For the sake of simplicity, regulations will be limited to European Union directives.

5.1. Software

General data protection regulation

The *General Data Protection Regulation (GDPR)* [24] is a directive that went into effect on May 25 of 2018 and sets out a series of rules and requirements for how organizations that handle personal data must collect, use, and protect that data, obtaining consent from individuals before. Overall, it is designed to give individuals more control over their personal data and to establish a more consistent and robust framework for protecting personal data in the EU.

Cyber resilience act

The *Cyber Resilience Act (CRA)* [25] is a directive that is proposed to take effect by mid 2023 and aims to safeguard consumers and businesses buying or using products or software with a digital component. Overall it will create beneficial conditions allowing users to take cybersecurity into account when selecting and using products with digital elements. Manufacturers will need to implement a coherent cybersecurity framework, facilitating compliance for hardware and software producers throughout a product's life cycle.

5.2. Hardware

Restriction of hazardous substances

The *Restriction of Hazardous Substances (RoHS)* [26] is a directive that went into effect on July 1 of 2006 and restricts the use of certain hazardous substances, such as lead, mercury, or cadmium, in the production of electrical and electronic equipment, protecting human health and the environment.

Electromagnetic compatibility

The *Electromagnetic Compatibility (EMC)* [27] is a directive that went into effect on January 1 of 1996 and aims to ensure that electrical and electronic equipment does not generate or emit excessive levels of electromagnetic interference (EMI), and that it is able to withstand electromagnetic disturbances from other equipment and sources. This helps to ensure that equipment operates in a safe and reliable manner.

Radio equipment directive

The *Radio Equipment Directive (RED)* [28] is a directive that went into effect on June 13 of 2016 and sets out the essential requirements that radio equipment must meet in order to be placed on the EU market. It is similar to the previous EMC. Overall, the RED is designed to ensure that radio equipment used in the EU is safe and performs as expected, facilitating the free movement of radio equipment within the EU and protecting consumers and other users from harmful emissions and other potential hazards.

Conformitè europëenne

The *Conformitè Europëenne (CE)* [29] is a directive that went into effect on July 1 of 1992 and sets out the conditions under which certain products can be marked with the CE mark. The CE mark indicates that a product meets the essential requirements of EU health, safety, and environmental protection legislation by complying with the EMC, LVD, TSD, RoHS or WEEE directives among others if applicable. Overall, the CE Marking Directive is designed to facilitate the free movement of goods within the EU and protect consumers and other users of CE-marked products from potential health and safety risks.

Waste electrical and electronic equipment

The *Waste Electrical and Electronic Equipment (WEEE)* [30] is a directive that went into effect on February 13 of 2003 and sets out the rules for the collection, treatment, recycling, and disposal of waste electrical and electronic equipment. Overall, the WEEE Directive is designed to reduce the environmental impact of waste electrical and electronic equipment by promoting the collection, treatment, recycling, and disposal of WEEE in a safe and environmentally-friendly manner.

General product safety directive

The *General Product Safety Directive (GPSD)* [31] is a directive that went into effect on December 3 of 2001 and aims to ensure that all products placed on the market within the EU are safe for consumers. As the GPSD generally applies to all products, it has a list of harmonized standards which cover different types of products. In this case, the electronics standards are the EN 60065 and EN 60950. The manufacturers must provide clear and accurate information on the safety of their products, including warning labels and instructions for use and provide a framework for the recall of unsafe products

Common charger directive

The *Common Charger Directive (CCD)* [32] is a directive that went into effect on June 29 of 2022 and aims to reduce the environmental impact of electronic devices by standardizing their charging capabilities and making USB-C charging port mandatory for a whole range of electronic devices. As seen later on in the *hardware components* section the product prototype does indeed have a USB-C port. It must be noticed, however, that it was not with the intention to follow this directive.

6. Knowledge Integration

I had the capability of successfully developing and finishing this Thesis project thanks to my past experience and the past years I have studied at the UPC FIB, where I had the opportunity to learn from scratch new skills and improve the existing ones. All of the subjects I took at university helped me indirectly with being able to create this product prototype, but I would like to mention some specific ones, and talk about how they aided me, as well as the specialization in computer engineering.

Moreover, it is expected the final outcome of the bachelor's Thesis to be a summarization and congregation of all the technical skill sets and competences the degree aims to offer to the student. Therefore, it is important to remark what technical competences this thesis develops.

6.1. Informatics engineering degree

As said, all the subjects have helped me through the development of the Thesis, ones more directly than others like building blocks that come together to form a robust wall.

Direct contributors can be separated in five different categories:

General programming and algorithms concepts
Although I already had past programming experience before entering the degree, *Programming 1* (PRO1), *Programming 2* (PRO2) and *Data Structures and Algorithms* (EDA) are three subjects that give you another perspective about coding and obviously several new concepts and algorithmic building blocks that helps anyone writing code in every single key press.

Low level computer knowledge
I would like to acknowledge all common subjects on the computer engineer branch, but it is probably *Introduction to Computers* (IC), the first subject on the hardware path that serves as a shocking introduction to how microcontrollers and computers in general work; and *Computer Interfacing* (CI), that introduces and goes deeply to the mainly used communication protocols in microcontrollers, several of which I had to use in this Thesis, such as UART[13], SPI[14] or JTAG[15] to name a few; the two compulsory subjects of the common phase of the degree that helped me the most on the development of the Thesis.

Project management comprehension
This Thesis is basically composed from two different domains: a practical one and a more management one. While all the other subjects mentioned help more on the development of

[13] Universal Asynchronous Receiver-Transmitter.
[14] Serial Peripheral Interface.
[15] Joint Test Action Group.

the former one, on the project management realm *Project Management* (GEP) and *Programming Projects* (PROP) are undoubtedly the most helpful courses of the degree.

What's more, in fact, although GEP is a short course spanning through the first month of the Thesis development, the finely arranged environment helps you write almost a third part of the whole Thesis document just at the start.

Database notions
Although it can be argued that databases are not a main part of this Thesis, it is true that having the necessary concepts about databases was a must for several proposed requirements and objectives such as being able to create users or schedulements of actions on actuators. That's why *Databases* (BD) was also important for the successful completion of this project.

Parallelism and concurrent programming theory
As the brain of this product prototype is a two-core microcontroller understanding the basics of parallelism and concurrent programming with the problems they carry and the possible solutions was crucial for the whole firmware part of this Thesis. *Parallelism* (PAR) was a compulsory course that allowed me to gain this important knowledge.

6.2. Computer engineering major

This is without any doubt a computer engineering Thesis, in fact a very ambitious one. The knowledgements of the specialization are employed everywhere in the Thesis, from the component selection and electrical schematic design to the printed circuit board design and all the low level firmware needed for the communication with the different hardware modules and the construction and configuration of the real time system that is the product prototype.

However, thanks to the wide objectives of this Thesis, it surely touches concepts about other specializations or even different degrees.

For instance, all the firmware necessary for the CRUDs and the HTTP server is directly related to the software engineering major. Moreover, the aim to touch almost all aspects of a retail electronic product also introduces concepts from other degrees such as the design and manufacturing of the casing and packaging, that could be related to mechanical and product design degrees.

As mentioned before, there were several common compulsory subjects that aided me in the development of the Thesis. However, there are three specific subjects from the computer engineering major that were the most influential ones without any doubt.

Design of microcomputer-based systems
DSBM is a subject that really lays the foundations on electrical designs and takes even further the concepts of hardware communication protocols seen in CI. Additionally it allows

the student to be able to make correct assumptions and decisions when selecting and comparing hardware components.

It was key to the selection of components and to the correct electrical schematic design of the product prototype.

Real-time systems
The STR course gives the student the foundations and knowledge about real time systems in not only a theoretical way but in a practical way. All the deliverables done in the subject helps you understand how microcontrollers are professionally programmed and controlled in the real world.

It was crucial on all the firmware parts of the product as the microcontroller used works and is programmed on a real-time operating system.

Computer engineering project
PEC is the subject. It is the most important course I took in the degree and computer engineering major for this Thesis with difference. It is in its essence a Thesis project in miniature with the great help and vast knowledge of the professors.

It was essential on every single step of this Thesis. Without taking PEC I strongly doubt I would have had the courage to do such an ambitious project. Specifically, it helped me design and manufacture the PCB of the product.

6.3. Technical competences

It was compulsory that the Thesis outcome was a project that greatly illustrates several technical competences of the informatics engineering degree, specifically about the computer engineering major. This resulting project exhibits seven different competences in depth and concurs with the considered competences on the project inscription. Moreover, an eighth one is also included, CEC2.2, and it is possible that other competences not contemplated also relate to this Thesis due to its ambition.

CEC1.1. To design a system based on a microprocessor/microcontroller. [In depth]

Neither to say that the core of the product prototype is an Espressif microcontroller that communicates with the other components on the system through SPI, TTL or PWM and with the user via SoftAP and the Internet.

The custom designed system schematic seems simple on the outside, yet it is a complex combination of selected hardware components that come together flawlessly to form the product prototype. Moreover, a custom working PCB was designed and even manufactured.

CEC2.1. To analyze, evaluate, select and configure hardware platforms for the development and execution of computer applications and services. [In depth]

For the development of this product prototype the different alternatives for the main microcontroller have been thoroughly analyzed and evaluated, as seen in the *hardware components* selection section, choosing the one that has been considered best, according to the criteria and decisions taken, which have been shown in this Thesis.

The microcontroller serves as the hardware platform on which all the other parts of the system are built and has been configured to be efficient but simple to develop upon.

The built system has two computer applications. The firmware on the microcontroller is a backend application or service. The web dashboard is another application, in this case it is the frontend of the system.

CEC2.2. To program taking into account the hardware architecture, using assembly language as well as high-level programming languages. [In depth]

As seen in the *firmware* section, there were several options in order to program the microcontroller. The decision taken was to use the ESP-IDF environment that allowed for a more low level development and unlocked several features critical for the correct completion of the product prototype.

Moreover, as the hardware platform is a microcontroller, programming taking into account the underlying architecture is a must. What's more, several specific elements of the architecture were directly used such as internal timers, interrupts, cryptographic accelerators and low level communications protocols.

CEC2.3. To develop and analyze software for systems based on microprocessors and its interfaces with users and other devices. [In depth]

The product prototype packs, really, a bunch of software, both direct firmware and web programming. As seen in the *firmware* section, a lot of use cases have been developed, always analyzing its efficiency and overall security.

Neither to say that, exactly this Thesis project does indeed interface with users and other devices, as it is an IoT gateway that allows end users to monitor and control their sensors and actuators via 433MHz communication frequency band.

CEC2.4. To design and implement system and communications software. [In depth]

Again, following the previous competence, this Thesis project in its essence is a 433MHz gateway device. Therefore, a system for 433MHz duplex communication, that is both transmission and reception, had to be designed and not only implemented theoretically but carried to the real world.

As seen in the *firmware* section, this RF communication software was even designed and developed from scratch, through the analysis, reverse-engineering and finally abstracting all the distinct communication protocols the different sensors and actuators used.

CEC3.1. To analyze, evaluate and select the most adequate hardware and software platform to support embedded and real-time applications. [In depth]

As said in the competence CEC2.1, for the development of this project the different alternatives for the hardware components have been thoroughly analyzed and evaluated, as seen in the *hardware components* selection section.

Moreover, as said in CEC2.2, the different software platform options were also analyzed and evaluated, as seen in the *firmware* section, finally selecting the ESP-IDF programming environment.

Neither to say that this product prototype is indeed an embedded real-time application. The system operates through a microcontroller using FreeRTOS as the real-time operating system.

CEC3.2. To develop specific processors and embedded systems; to develop and optimize the software of these systems. [In depth]

Undoubtedly, as already seen in all the previous competences, a lot of software was developed for the Thesis project, which again, is an embedded system, and it was always done with optimization and efficiency in mind.

CEC4.2. To demonstrate comprehension, to apply and manage the guarantee and security of computer systems. [In depth]

Security is one of the main three pillars of this Thesis as seen in the *requirements* section. Without a doubt, this product prototype was developed and built with security in mind. Several techniques and technologies were analyzed and applied to manage and guarantee the security of this embedded computer system such as HTTPS, secure boot, flash encryption, TLS or AES encryption algorithms to name a few.

7. Initial Research

As stated everywhere in this Thesis, the intrinsics of the project is the 433MHz radio frequency band. The core objective is to be able to receive data from sensors and control actuators operating in that frequency. As such, the first thing to do is to research how to communicate with such devices in order to assert that the project is even feasible.

7.1. Test subjects

The test subjects selected are common devices sold using this technology and consist of 1 sensor and 2 actuators:

a) Open-close alarm. This device uses a Hall-effect[16] sensor to trigger the sending of an RF signal when both parts are close together or separated.

Figure 5: Splenssy open-close alarm.
Source: Compiled by the author.

b) Wireless socket. This device uses a relay[17] to switch off or on any other appliance that is plugged in on its socket and its triggered upon the receival of an specific RF signal.

Figure 6: SatChef wireless socket.
Source: Compiled by the author.

c) Wireless motorized blinds. This device uses an electric motor to roll up or down its window blinds and is also triggered upon the receival of an specific RF signal.

Figure 7: Master Otello wireless motorized blinds.
Source: Compiled by the author.

[16] Type of sensor which detects the presence and magnitude of a magnetic field.
[17] Electrically operated switch.

7.2. Communication protocol

After a long investigation on several Internet resources, literature and an extensive analysis of RF signals of different devices, it can be concluded that there is not a common protocol used in the communication of such devices.

That means, every selected device must be analyzed and its communication protocol reverse-engineered one by one. All this research must conclude in the understanding, decoding and a final generic abstraction that oversees the different protocols.

Frequency analysis

As already mentioned in the *methodology* section, two programs have been used to perform the signal analysis: *Gqrx* [33] and *Universal Radio Hacker* [34]. Moreover, an *RTL-SDR* [35] radio frequency scanner has been used to receive and demodulate the RF signals.

The following *Figure 8* depicts the radio frequency heatmap, gathered using GQRX, of the three test devices at the same time. Warmer colors denote more signal intensity in that specific frequency. Colder colors, on the other hand, means less signal intensity. As expected, the signals from the open-close alarm (OCA), wireless socket (WS) and the wireless motorized blinds (WMB) are more intense around the 433.92MHz frequency, because all three devices work, indeed, in the 433MHz band. The other signals seen are duplicated mirages of the former caused by their natural components or directly environmental interferences.

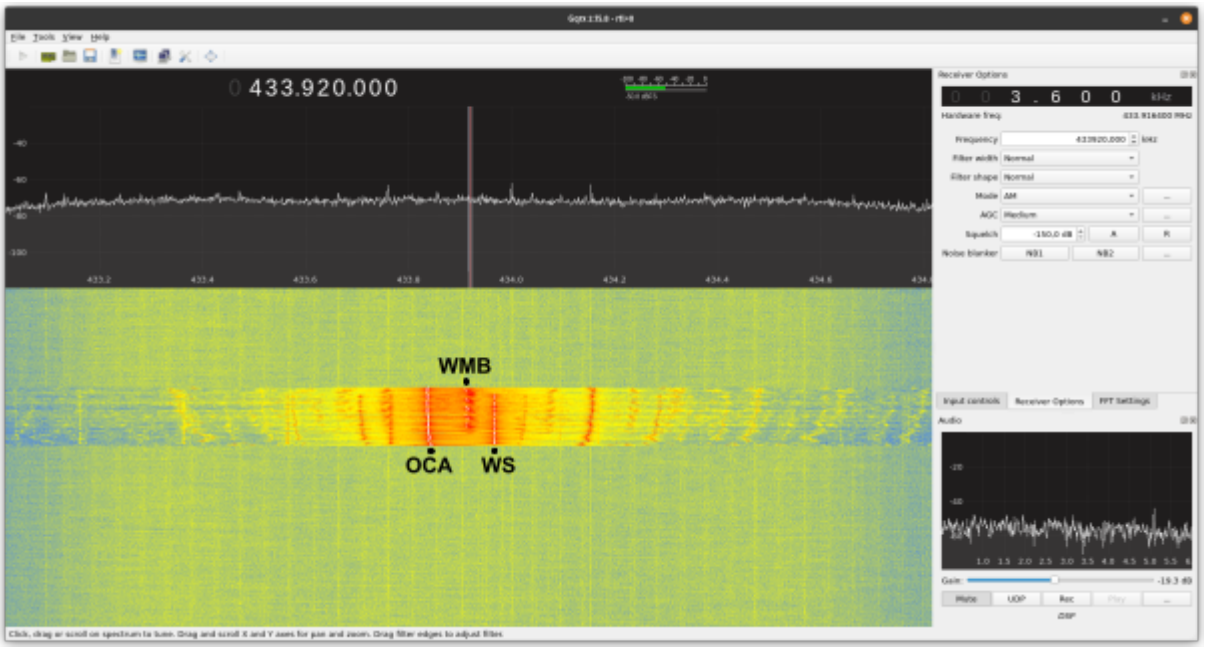


Figure 8: Radio frequency heatmap of the test subjects. Marked by the initials “OCA” refers to the open-close alarm, “WMB” refers to the wireless motorized blinds and “WS” refers to the wireless socket. *Source: Compiled by the author*.

The previous gathered frequency parameters of the test subjects can then be used to scan their signals and perform a more in depth signal analysis.

The following *Figure 9* contains the scanned signals, gathered using URH, of three pulsations of the "up" button on the wireless motorized blinds controller. At this stage, the decoded data is very coarse, it is just a time collection of the thousands and thousands samples collected by the RTL-SDR. In order to be able to infer more information, parameter tuning is required.

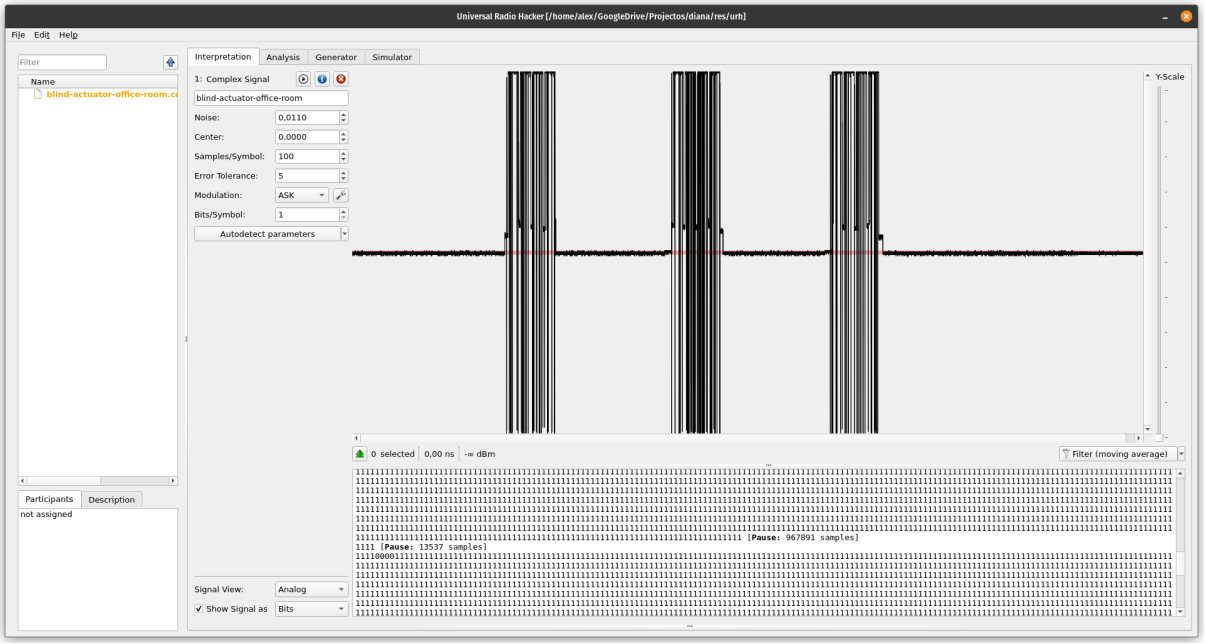


Figure 9: Scanned signals graphic of the wireless motorized blinds controller.
Source: Compiled by the author.

Modulation analysis

Besides noise degree, center level and error tolerance, which are parameters automatically tuned by the software, the other main parameters are modulation and samples per symbol (bit). Thanks to a prior investigation on the documentation about the test devices it is known that they work via digitally amplitude modulated signals, and thus the former parameter can be set to ASK. Doing so, clears up the signal, revealing, on the following *Figure 10*, what appears to be four times the "up" command signal sent by the controller to the window blinds on every button pulsation.

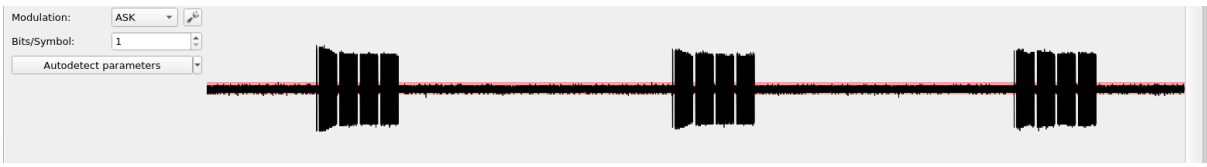


Figure 10: Scanned signals graphic of the wireless motorized blinds controller using ASK modulation. *Source: Compiled by the author*.

Phase analysis

Zooming in on several of those signal commands we can start to see patterns that let glimpse some protocol details. The following *Figure 11* depicts the decomposition of the "up" command signal, consisting of three phases. From now on, I will be using the word "packet" to refer to both actuator command and sensor data signals.

Figure 11: Decomposition of the "up" command signal of the wireless motorized blinds controller. *Source: Compiled by the author*.

a) Sync. This first phase marks the beginning of a new packet. It consists of a short pulse followed by a predefined, very long gap. It helps the receiver to prepare its internal clock syncing it with the pulse timings.

b) Preamble. The following phase consists of a burst of short pulses followed by a long gap. It helps the receiver to tune its error tolerance parameters that define the base pulse width.

c) Data. The final phase contains the data sent itself. Data is sent as a sequence of bits encoded on pulses and gaps, the width of which are multiples of the base pulse width.

Moreover, as it can be seen at the end of the graphic, it must be noticed that the very last short pulse is not part of the data but it is the start of the sync phase of the next packet signal. The following *Figure 12* shows so.

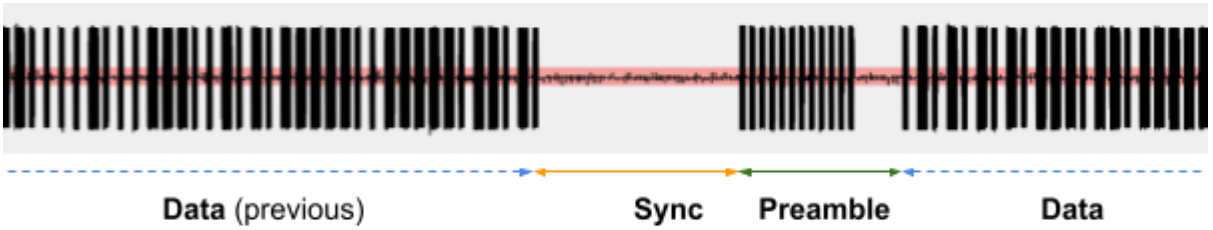


Figure 12: Mixed decomposition of two "up" command signals of the wireless motorized blinds controller. *Source: Compiled by the author*.

Data analysis

As said, the data phase contains the sent bits encoded on pulses and gaps the width of which are multiples, or nth-divisions, of the average measured pulse width, or base pulse width, that is 400µs in this case.

Taking a closer look at the samples it can be concluded that there are indeed two patterns, as it can be seen in the following *Figure 13*.

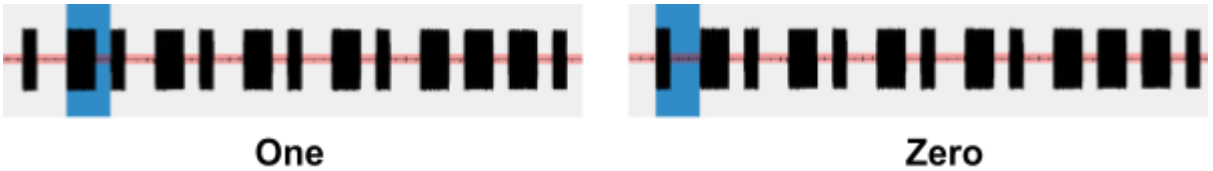


Figure 13: Data pattern decomposition of the "up" command signal of the wireless motorized blinds controller. On the left the pattern for a "one" bit and on the right for a "zero" bit. *Source: Compiled by the author*.

The left pattern consists of a pulse of two divisions of the base width and a gap of one division. The right pattern on the opposite consists of a pulse of one division of the base width and a gap of two divisions. Both patterns span a total of three divisions, that is 1200µs.

If we assume that having a larger pulse width in relation to the gap means that it is probably a one and that having a larger gap width in relation to the pulse is probably a zero and that the samples per symbol (bit) parameter totals to 1200, we end up with the decoded bits of the packet as it can be seen in the following *Figure 14*.

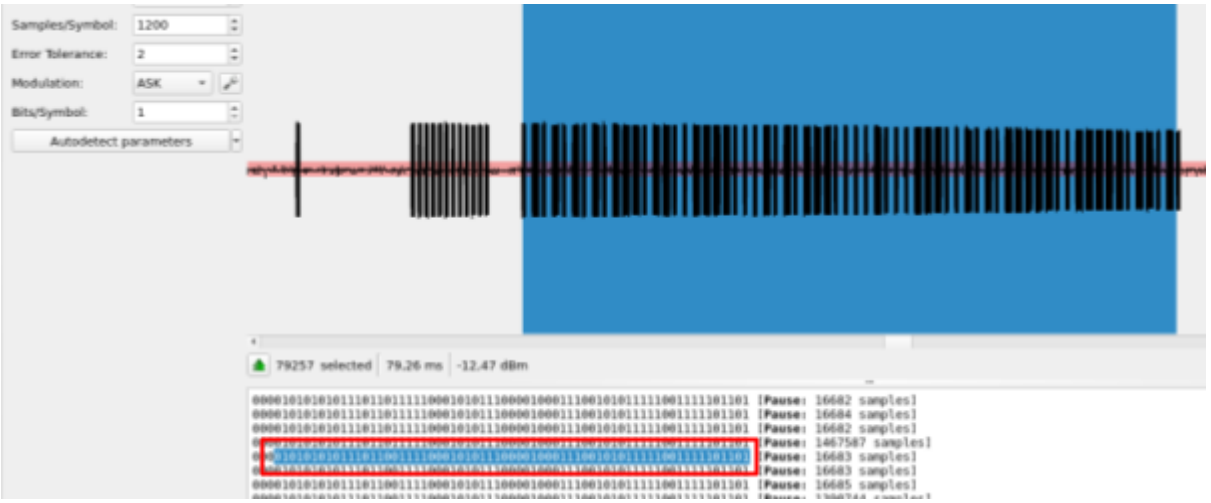


Figure 14: Decoded bits of the "up" command signal of the wireless motorized blinds controller (marked by a red rectangle). *Source: Compiled by the author*.

It is a 64 bit-long number, further confirming our hypothesis. The communication protocol for the wireless motorized blinds has been successfully reverse-engineered.

The same research technique has to be applied on the other buttons of the wireless motorized blinds controller.

On the open-close alarm, as seen by the summarization in the following *Figure 15*.

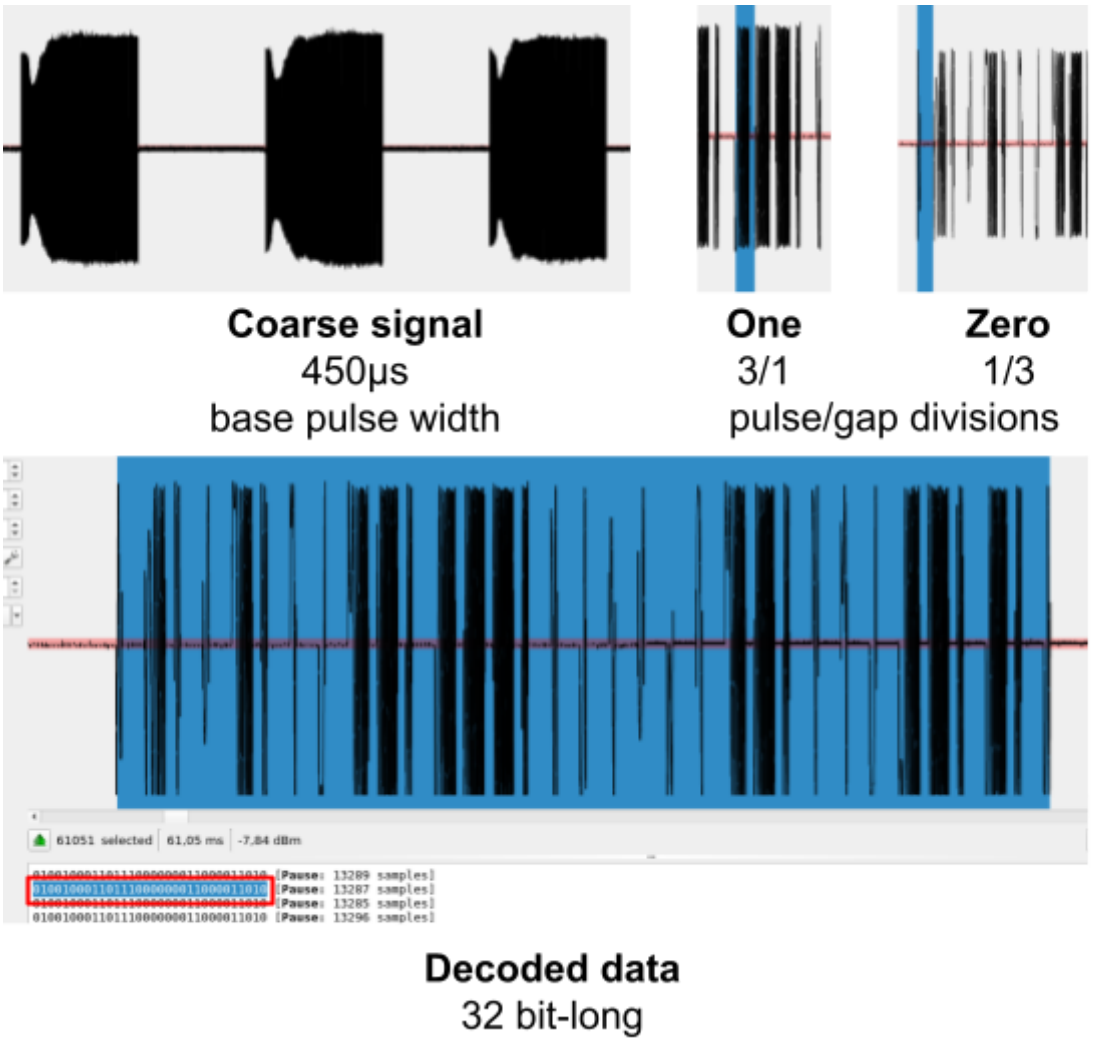


Figure 15: Summarized communication protocol analysis of the open-close alarm.
Source: Compiled by the author.

And on the wireless socket, as seen by the summarization in the following *Figure 16*.

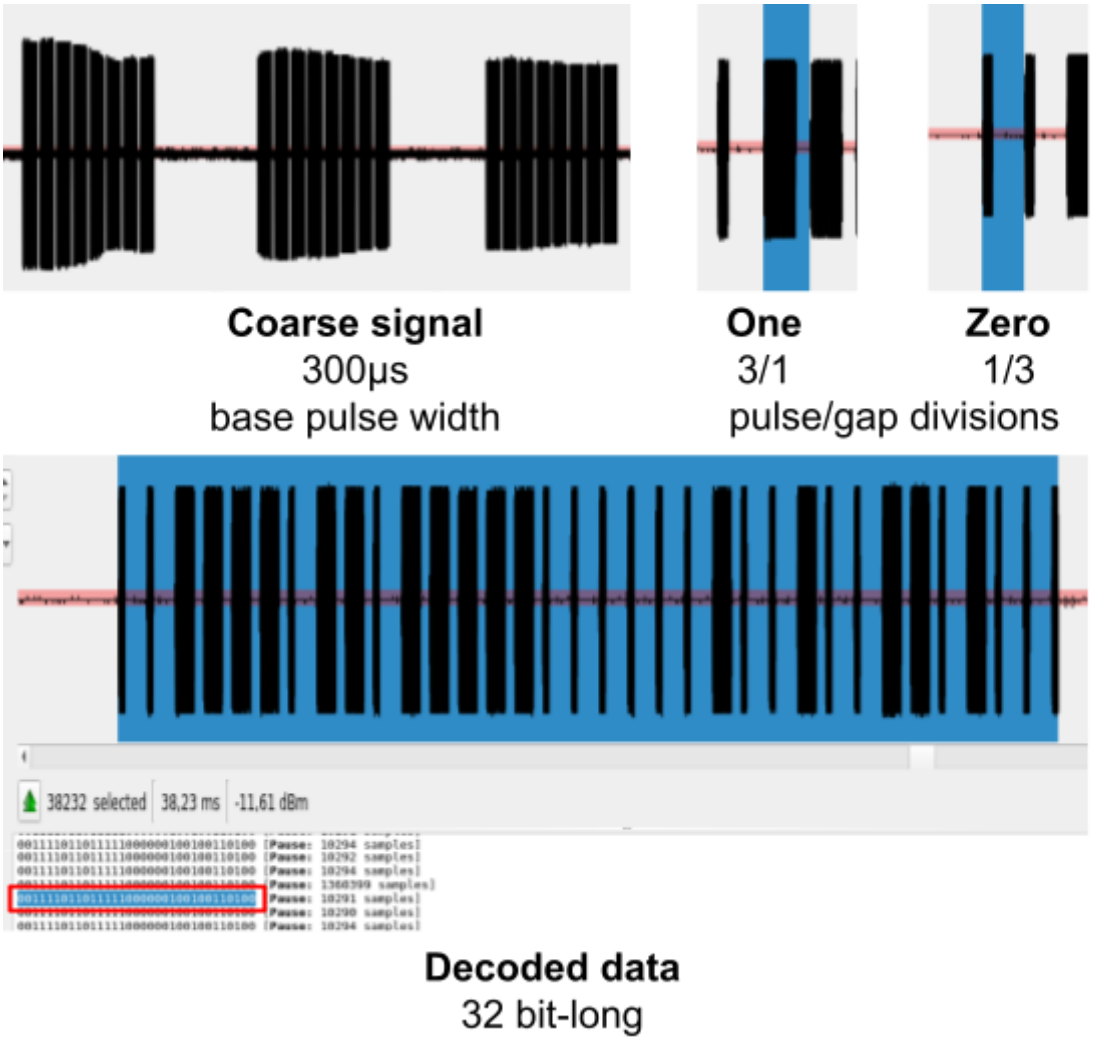


Figure 16: Summarized communication protocol analysis of the wireless socket controller.
Source: Compiled by the author.

As already predicted at the start of this section, the communication protocols used by the test devices are different, not only varying on the signal or data parameters, but even on the phases.

The windows blinds controller has a “preamble” phase that neither the open-close alarm or wireless socket controller has.

Moreover, doing a fast research on other devices protocols, it can be seen that the length or form of both “sync” and “preamble” phase is not the same on all. The pattern on the data phase, however, seems to follow the same rule on all protocols, differentiating ones and zeros by the pulse to gap width ratio.

Results

The decoded actuator commands or sensor state identifiers are as follow:

a) Open-close alarm:
   i) "Open" state identifier: $0x48DC061A$
   ii) "Closed" state identifier: $0x48DC0609$

b) Wireless socket:
   i) "On" command: $0x3DBE049A$
   ii) "Off" command: $0x3DBE0892$

c) Wireless motorized blinds:
   i) "Roll up" command: $0x557607EE11CAF9F6$
   ii) "Stop" command: $0x55797BEE11CAF9F9$
   iii) "Roll down" command: $0x557D3BEE11CAF9FD$

8. Taken Approach

8.1. Proposed system

As said at the beginning of the document, a hypothetical user must be able to control her/his sensors and actuators remotely through the Internet. That cannot be done only in software, a hardware platform is required capable of receiving and transmitting data to those 433MHz devices and has to be connected somehow to the Internet to be accessible for the end user.

At its core, the proposed system is an electronic device formed by a Wi-Fi-capable microcontroller, running a light-weight real-time operating system, with a pair of 433MHz-tuned antennas along with a transmitter and receiver modules.

Below, in *Figure 17*, there is a simplified overview of the system. Multiple users, either on mobile or desktop devices, interacturate with the hardware product through the Internet, that is, outside from their home's Wi-Fi network; or through an intranet, that is, inside of their home's Wi-Fi network; or directly using the network created by the product itself. The presented interface is a self-hosted web application. Then, the product interactuates with the 433MHz devices through the transmitted or received signals using the specific protocol required for each one, as has been already shown in the last section *initial research*.

Moreover, the product can maintain the time for the trigger scheduler thanks to a built-in real-time clock module with an Internet clock synchronization. The product is powered by a plug using an already existing cable and charger through an USB-C port.

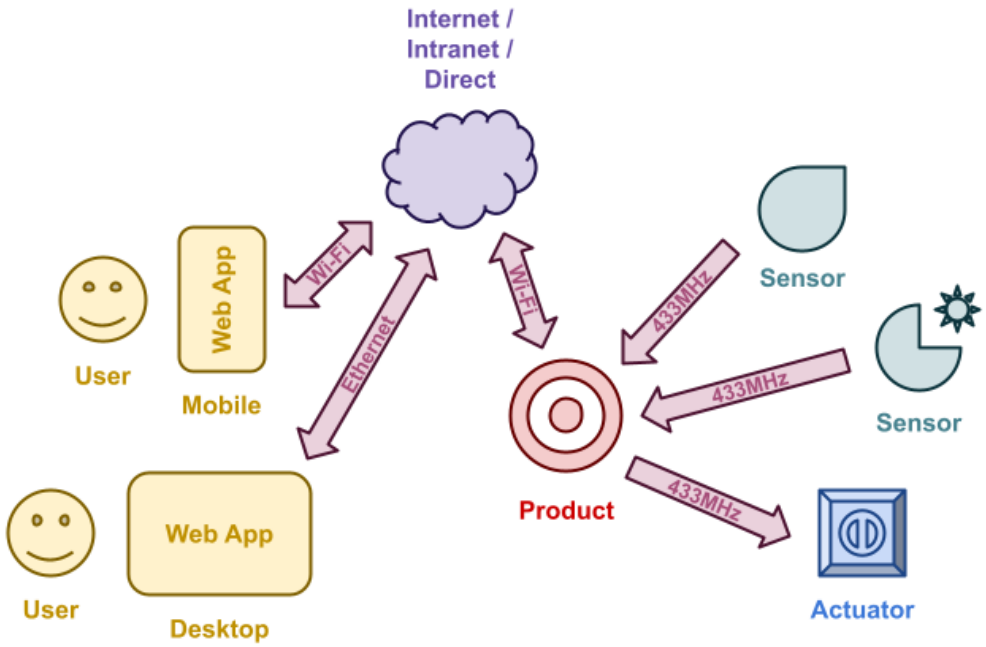


Figure 17: Conceptual diagram of the system. *Source: Compiled by the author*.

8.2. Decisions taken

Before brainstorming any ideas or starting to design any part of the product, several decisions were taken to put on track general aspects, obviously respecting the previous set requirements and objectives in the *scope* section.

a) The "code name" of the product is "Diana Dot" and the aesthetic of the electronic device must resemble a bulls-eye. These are important as they will determine the casing and packaging designs later on.

b) The electronic device must not be bigger than the palm of a hand, the end user should be able to grab it one-handed. This is more an arbitrary decision, in the end the device does not have size limit requirements, although reducing its size as small as possible without sacrificing on anything is beneficial for hypothetical storing and transportation.

c) The electronic device will be powered through a USB-C port using standard 5V smartphone chargers and must consume 1A max.

d) To simplify the electronics, it is not necessary that the device preserves the time on a power reset, as its main usage happens when it is connected to the Internet and thus clock synchronization can be used. That does not mean that the RTC module is useless, it will maintain the time when Internet connection is down to avoid feature usage limitation, as it was one of the requirements set.

e) The electronic device will include a visual cue that indicates that it is on and connected to the internet, giving additional feedback to the user on certain situations. Such cue must be visible at all times.

f) The initial Internet credentials provisioning must not use any other technology than Wi-Fi. In this case the product will use a self-created network running a captive portal to which the user can connect to.

g) To comply with the "off-grid" requirement, the web application will not be hosted on a third party server, the electronic device must itself store and serve the application.

h) The device firmware will be written in C (not Arduino flavor) using the microcontroller vendor toolchain to be able to access all of its low-level APIs efficiently. As seen later on the *firmware* section, C++ will be used a little to order the code using classes with the singleton[18] and dependency injection[19] pattern.

[18] Software design pattern that restricts the instantiation of an object to a singular instance.
[19] Software design pattern in which objects are instantiated through the objects that it depends on.

8.3. Conceptual diagrams

Several simplified diagrams were created before designing each aspect of the product. Having a conceptual view on what is required and how it should look is essential not to get out of the path and be consistent in all of the final designs. Later on the Thesis, each specific section includes several CAD designs and 3D renders to visualize more in depth each aspect of the built product prototype.

Schematic

The following *Figure 18* depicts the simplified conceptual diagram for the electrical schematic. Each box represents a required hardware component and each line a connection.

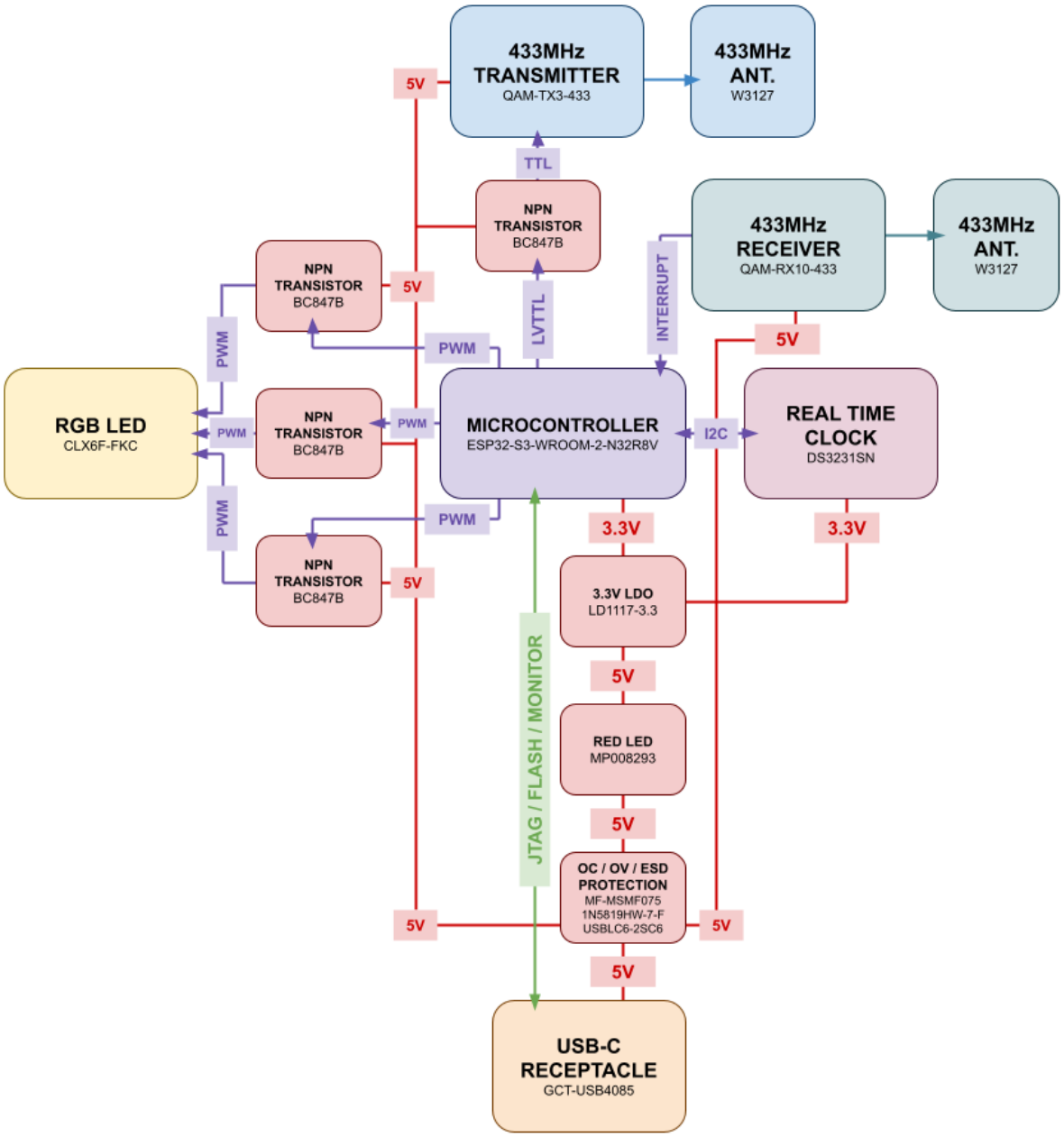


Figure 18: Conceptual diagram of the schematic. *Source: Compiled by the author*.

The complete information for each hardware component and a complete BOM is presented later on the Thesis document in its own section. Detailed and exact circuits, connections and component values can be found later on the *schematic* section.

The red connections represent the power lines required for the components. As decided, the power input of the system is 5V through a USB-C 2.0 receptacle. A voltage regulator is needed to feed the microcontroller and the real-time clock module which use 3.3V.

Four layers of protection circuits have been added to safeguard the device from issues of the connected power supply, computer or even the device itself:

a) Over-current (OC): protects the device from drawing too much power and potentially burning the internal components or even the PCB traces. This can happen when a component misbehaves, is broken or directly because there was a short-circuit. The circuit has been implemented using a polyfuse, which is a thermostatic resettable fuse.

b) Over-voltage (OV): protects the device from a sustained input of excess voltage. A long exposure can destroy the internal components. This can happen if a higher voltage than the used internally, 5V, is provided through the power source. The circuit has been implemented using a schottky rectifying diode.

c) Reverse bias (RB): protects the external power supply or computer from power flowing in the other direction that can lead to OV situations in the connected end. This can happen when a component misbehaves, is broken or directly because there was a short-circuit. The circuit has been implemented using a schottky rectifying diode.

d) Electrostatic discharge (ESD): protects the device from a sudden transient huge spike of voltage (from 500V to upwards 8,000V) that is applied somewhere in the circuit leading to the destruction of the internal components. This can happen by static electricity from the exposed USB-C port. The circuit has been implemented using an integrated circuit composed of array diodes.

Several NPN transistors have been used to step up the current needed for the RGB LED and the transmitter module. Moreover a red LED has been included to indicate that the system is receiving power.

The microcontroller interfaces with the computer through JTAG over the USB-C 2.0 so code can be programmed (flashed) on it. Moreover, on the same JTAG line logging output can be monitored through a command line. The selected microcontroller is also capable of being debugged on-site using JTAG hardware breakpoints.

In order to interactuate with the real-time clock module the microcontroller uses I2C. 433MHz signals are received on an interrupt-basis and the transmitter is controlled digitally

through a single pin. It must be noticed that the receiver is directly connected although it uses 5V logic because the selected microcontroller's GPIO pins are 5V capable. Finally the RGB LED is controlled via PWM[20] on each color channel.

As seen, both 433MHz modules are directly connected to a frequency-tuned antenna. Having two antennas means that receiving and transmitting can be done in parallel (as seen later the microcontroller has two cores). This is extremely useful to assert that the transmitted command is correct.

Moreover, 3 grounded M2[21] mounting holes are included to allow mounting the PCB onto the casing later on.

The electrical schematic will be designed using *KiCad* [36].

PCB

The following *Figure 19* depicts the conceptual diagram for the 2-layer printed circuit board (PCB), where each box represents a principal hardware component and the green background the fiberglass substrate. It must be noticed that the diagram is a simplified version and thus its dimensions are not exactly proportional to the real scale. The full measures of the PCB and several CAD designs and 3D renders are included in the *PCB* section of this Thesis document.

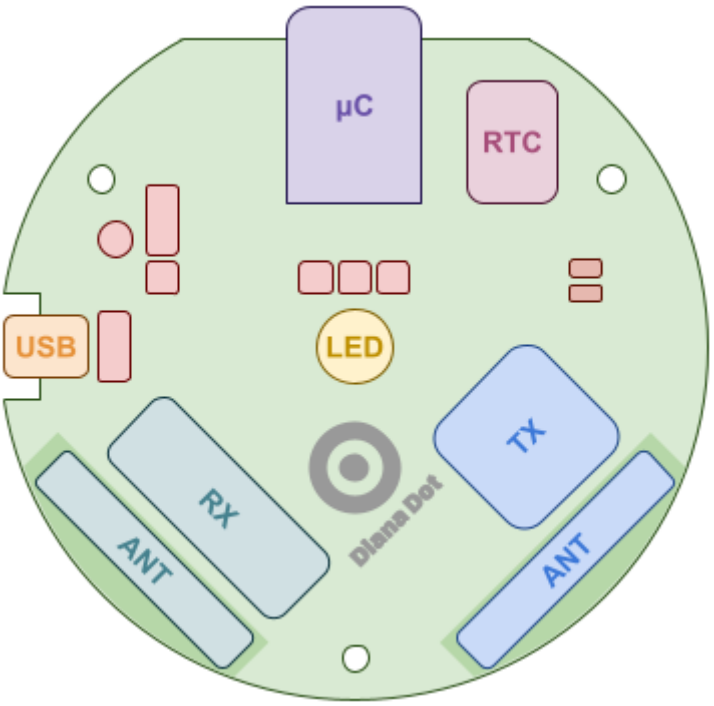


Figure 19: Conceptual diagram of the printed circuit board. *Source: Compiled by the author*.

[20] Pulse Width Modulation.
[21] ISO metric screw thread size of 2mm diameter.

The shown part is only the top layer as the bottom layer is a copper pour to keep it as a clean ground plane without any placed component or routed traces, although a few were necessary. The top layer is not copper-poured to avoid potential crosstalk within the board traces.

As decided before, the casing will have the shape of a bulls-eye. To keep its size at its minimum, the PCB follows the same outline so it can be mounted directly to the casing. However, it's not circular all around. A low side profile was needed to narrow the casing size, that's why instead of mounting the USB-C receptacle standing out, it is placed inside the PCB and there is an indent so it can still be accessed.

Almost all of the selected components are SMD[22] mounted with the exception of the USB-C receptacle, the antenna pair and the 433MHz communication modules which are THT[23]. Because of the small component pads along that the chip has pads on its bottom side, the PCB was soldered using metallic solder paste with flux through an industrial electronic oven, more on that in the PCB *manufacturing* section of this Thesis document.

The microcontroller has been placed next to the top side edge to, specifically, lay out its antenna of the PCB to keep crosstalk at minimum or any other interference from the board, following the microcontroller's vendor hardware design guidelines. Again, through a cutout, so the overall shape remains a circle.

The other component placement has been mainly designed around the microcontroller requirements and used GPIOs pins, which reside on all the contour of the chip.

The RTC module is placed directly next to the used I2C pins on the right side of the chip, and so does the RGB LED and its transistors on the bottom side of the chip, which is also centered on the board so the light comes out of the center of the casing later on. It is worth mentioning, that although the RTC is referred to as a module, in reality it's an standalone chip soldered directly to the PCB along with its required circuit.

Moreover, two buttons have been also included to assist the firmware development process. One allows you to manually reset the microcontroller whenever it gets stuck without needing to suddenly disconnect the power source. The other one allows you to switch from firmware download mode (flashing) to firmware execution mode (booting). Although the USB JTAG interface has the ability to automatically switch from one mode to the other, it is useful as a fallback if it becomes stuck indefinitely in the download mode.

The USB-C receptacle, protection and converter circuits are placed on the left side of the microcontroller close to the PCB edge, all close together to reduce electrical resistance. The placement and routing of the USB 2.0 is especially critical. In order to ensure the correct functioning of the full-speed capabilities of USB 2.0, the data D+ and D- traces had to be

[22] Surface Mounted Device.
[23] Through-Hole Technology.

routed as a differential pair, keeping both tracks close and with the same length. Trace widths were also configured to ensure the differential characteristic impedance of 90Ω required in the USB 2.0 specification.

The 433MHz receiver and transmitter are placed separately and on the opposite side to the microcontroller to keep antenna coupling at minimum. Both components come as modules, that means they come with their own PCB. Pin headers have been used to solder them to the board.

Their antennas are as close to the modules and have an underneath ground plane clearance in order to keep PCB interference at minimum. Around them there is also a grounding fence formed by a grid of vias, and the antenna trace is configured to ensure the characteristic impedance of 50Ω following the antenna's manufacture hardware design guidelines.

Finally, the three M2 mounting holes are placed equidistantly from one another near the PCB edges forming a triangle. Moreover, the product name and logo have been printed onto the board solder mask by serigraphy.

Trace routing, properties, along with other PCB features and design requirements are explained in depth in the *PCB* section of this Thesis document.

The PCB will be designed using KiCad and manufactured by the German vendor *Aisler* [15].

Casing

The following *Figure 20* illustrates the conceptual diagram for the plastic casing. It must be noticed that the diagram is a simplified version and thus its dimensions are not exactly proportional to the real scale. On the left there is the side view, featuring the product on a table while a user is looking directly at it. On the right there is the top view, featuring some approximate measures and proportions. The full measures of the device and several CAD designs and 3D renders are included in the *casing* section of this Thesis document.

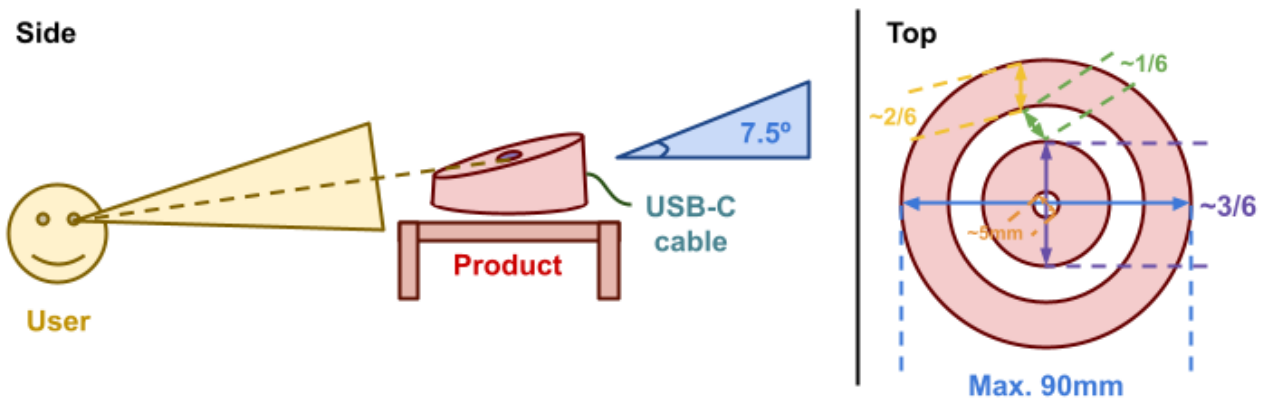


Figure 20: Conceptual diagram of the casing. *Source: Compiled by the author*.

It's a two-part casing, and as decided, the shape of it must resemble a bulls-eye, that's why the shape of the body is a cylinder terminated with a circular lid that mimics the outline of a bulls-eye. It has a central hole with a little diffusive plastic lens to let the light out from the RGB LED inside. The white section seen on the top lid view of the right side diagram will be negatively extruded downwards as the manufacturing mechanism used for the casing, 3D printing, does not allow for two colors. The PCB will be installed inside the casing and secured with three M2 screws.

On the side view we can see a user looking at the device. In order to make the status LED clear the casing is sliced in angle, tilted about 7.5º (general tilt of keyboards), so if the product is placed on a high surface the light can still be visible if looked directly. The other hole on the casing apart from the top one is one at the back. Its purpose is to expose the PCB's USB-C receptacle to the exterior so a cable can be connected through it.

The casing features a snap-fit mechanism that allows it to firmly close the lid into the body, consisting of two symmetric little extrusions inside opposite sides in the body, and two symmetric little indents inside opposite sides in the lid. A more detailed cross section analysis is included in the *casing* section of this Thesis document.

Finally, the material selected for the construction of this casing is a white semi-translucent PLA[24], which will allow easy manufacturing while maintaining strong and resistive properties. The white color is selected to have a good contrast with the light of the RGB LED, and making it translucent allows the light to better propagate and blur. The casing will be designed using *Fusion360* [37] and manufactured by a local 3D-printing facility.

Packaging

The following *Figure 21* illustrates the conceptual diagram for the cardboard packaging. It must be noticed that the diagram is a simplified version and thus its dimensions are not exactly proportional to the real scale.

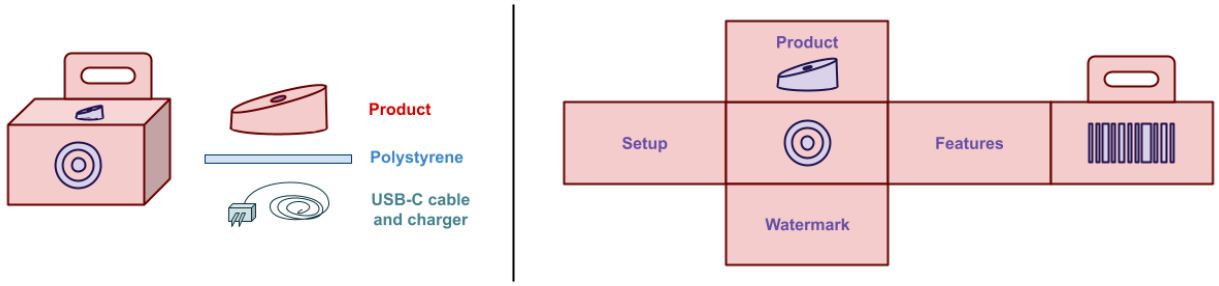


Figure 21: Conceptual diagram of the packaging. *Source: Compiled by the author*.

On the left there is the product package completely folded and a depiction of what's inside of it. On the right side there is the package dieline fully unfolded on all of its faces

[24] Polylactide.

corresponding to the left drawing. The full measures of the packaging and several CAD designs and 3D renders are included in the *packaging* section of this Thesis document.

As seen in the left part of the diagram the conceptuated package is a rectangular box with a hanging hole at the back. The box is opened from the top and features a flap-based closing mechanism. The proportions of the packaging are the smallest possible sizes required to include a USB-C cable and charger, a thin polystyrene separator and the product itself comfortably. Although it is common for electrical devices to come with instruction booklets, as stated before in the *scope* section any sort of compliance, such as user manuals, lay outside of the objectives of this Thesis as it is already a very ambitious one.

Different designs will be printed onto all faces of the packaging as seen in the right side of the diagram. The front face will feature the product logo (a bulls-eye) and the back face some electrical specifications and stamps along with a (fake) barcode with its assigned UPC[25] code.

The top face will be the most important one and includes a 3D render of the product along with its name and a descriptive tagline. The bottom face will just include a company watermark and the mounting mechanism to fold the package itself.

The left and right sides will be for product information. The left side will include the setup steps to onboard the user once the device is bought and the right side will show a list of features and its descriptions about the product itself.

Finally, the material selected for the construction of this package is a white coated 400gsm cardboard, which will allow easy manufacturing while maintaining sufficient strong and resistive properties. The packaging will be designed using *Figma* [38] and manufactured by the Italian vendor *Packly* [16].

[25] Universal Product Code.

Firmware

As previously decided the product's web application, or frontend, will be embedded and served directly from the device itself to avoid third-party dependencies. However, the development of both sides, backend and frontend, will be separated to allow modularity and different programming languages and environments. The both parts of the firmware will communicate through a REST API interface layer, composed of several requestable endpoints.

Backend

The following *Figure 22* depicts the simplified conceptual diagram for the backend code of the firmware.

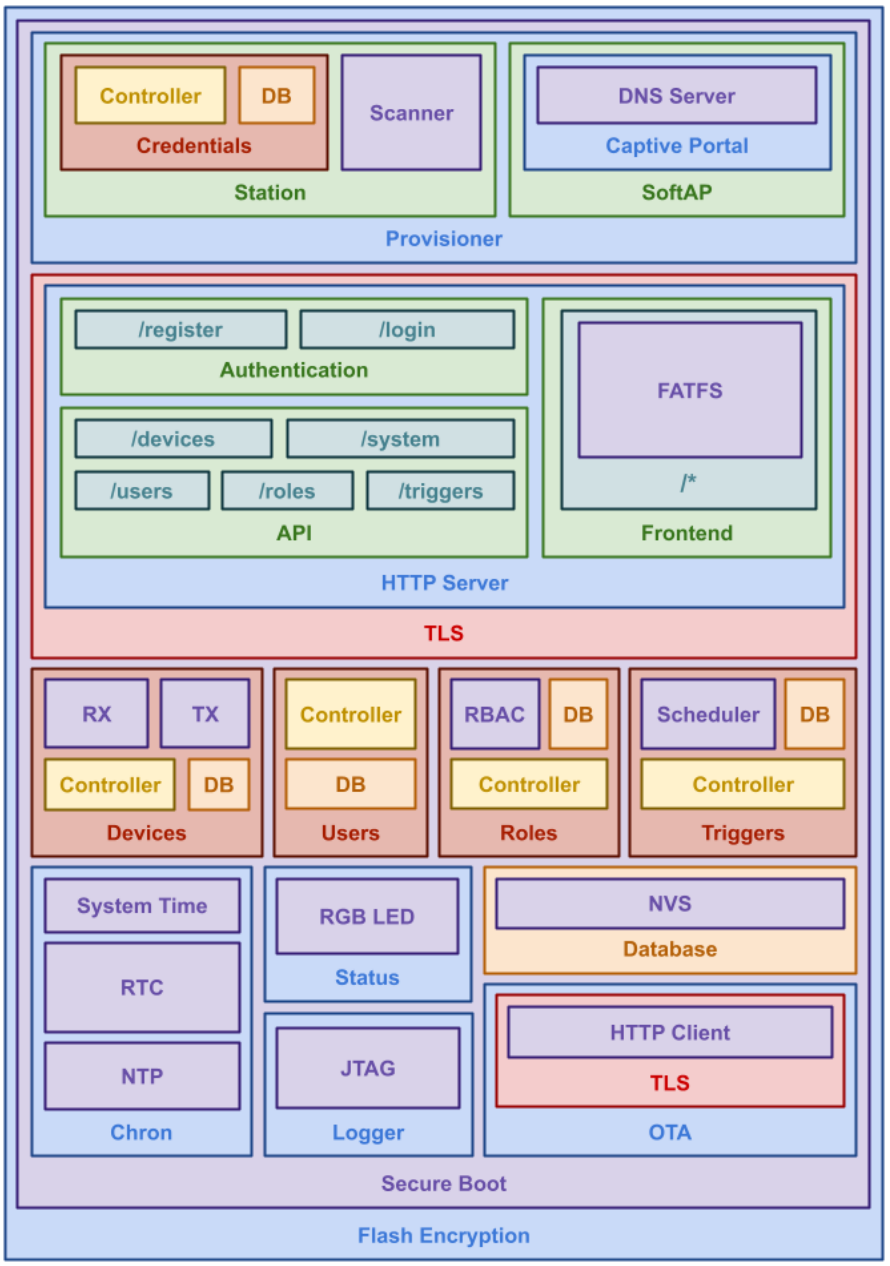


Figure 22: Conceptual diagram of the backend firmware. *Source: Compiled by the author*.

The codebase itself is very big and has several complex parts. A more detailed explanation can be found in the *firmware* section of this Thesis document. Each box represents a software component, technology or mechanism used, and inside of each there are included the sub-components that make up the former.

The backend has been implemented using the selected microcontroller toolchain and environment, that is the *ESP-IDF* [39] framework through the *VsCode* [40] development environment. The specific toolchain version used for this Thesis is “release/v5.0”. Although programming through the *Arduino* [41] IDE was also an option it was discarded because of the lack of access to several low level APIs such as cryptography algorithms, sockets, over the air updates, storage and deep chip configuration such as the partition table, flash encryption or watchdog triggers. The toolchain imposes the code to be written in plain C to access such APIs, but C++ will be used a little to order the code using classes with the singleton and dependency injection pattern.

Moreover, the framework allows an efficient use of both cores of the microcontroller by using a modified version of the real-time operating system *FreeRTOS* [42], therefore the system will be composed of a set of tasks running parallely and concurrently. *LwIP* [43] is used by the toolchain for the TCP/IP stack driver.

The following *Table 8* summarizes all the firmware tasks running in the system, either directly created by me or indirectly created by the framework. The table includes the task name, priority (a higher number means more priority), core (tasks can be unpinned), whether it's a built-in task and a short description. The table is ordered by priority descendingly.

NAME	PRIORITY	CORE	BUILT-IN	COMMENT
ipc0	24	0	YES	Inter-process communication signaling for core 0
ipc1	24	1	YES	Inter-process communication signaling for core 1
wifi	23	0	YES	Wifi driver
esp-timer	22	0	YES	High resolution hardware timer
sys-evt	20	0	YES	System events handler
tiT	18	0	YES	LwIP TCP/IP driver
capdns	16	ANY	NO	Captive portal DNS server
Provisioner	11	ANY	NO	Provisioner Wi-Fi connection retrier
Receiver	10	0	NO	Receiver packet manager
Transmitter	10	1	NO	Transmitter packet manager
httpd	9	ANY	YES	HTTP server
Chron	8	ANY	NO	RTC & NTP time synchronizer
Trigger	7	ANY	NO	Trigger scheduler
Status	6	ANY	NO	Status RGB LED controller
main	1	0	YES	Firmware main entrypoint
Tmr Svc	1	0	YES	Low resolution software timer
IDLE	0	0	YES	Inactive resource liberator for core 0
IDLE	0	1	YES	Inactive resource liberator for core 1

Table 8: Summary of the firmware tasks running on the operating system.
Source: Compiled by the author.

The core of the backend is created around the 5 models of the system: devices (sensors and actuators), triggers, users, roles and credentials as seen in brown on the previous *Figure 22* and in the following simplified UML diagram of *Figure 23*. It must be noticed that the types of the attributes and members has been simplified to a general language, for instance from "const char *" to "string".

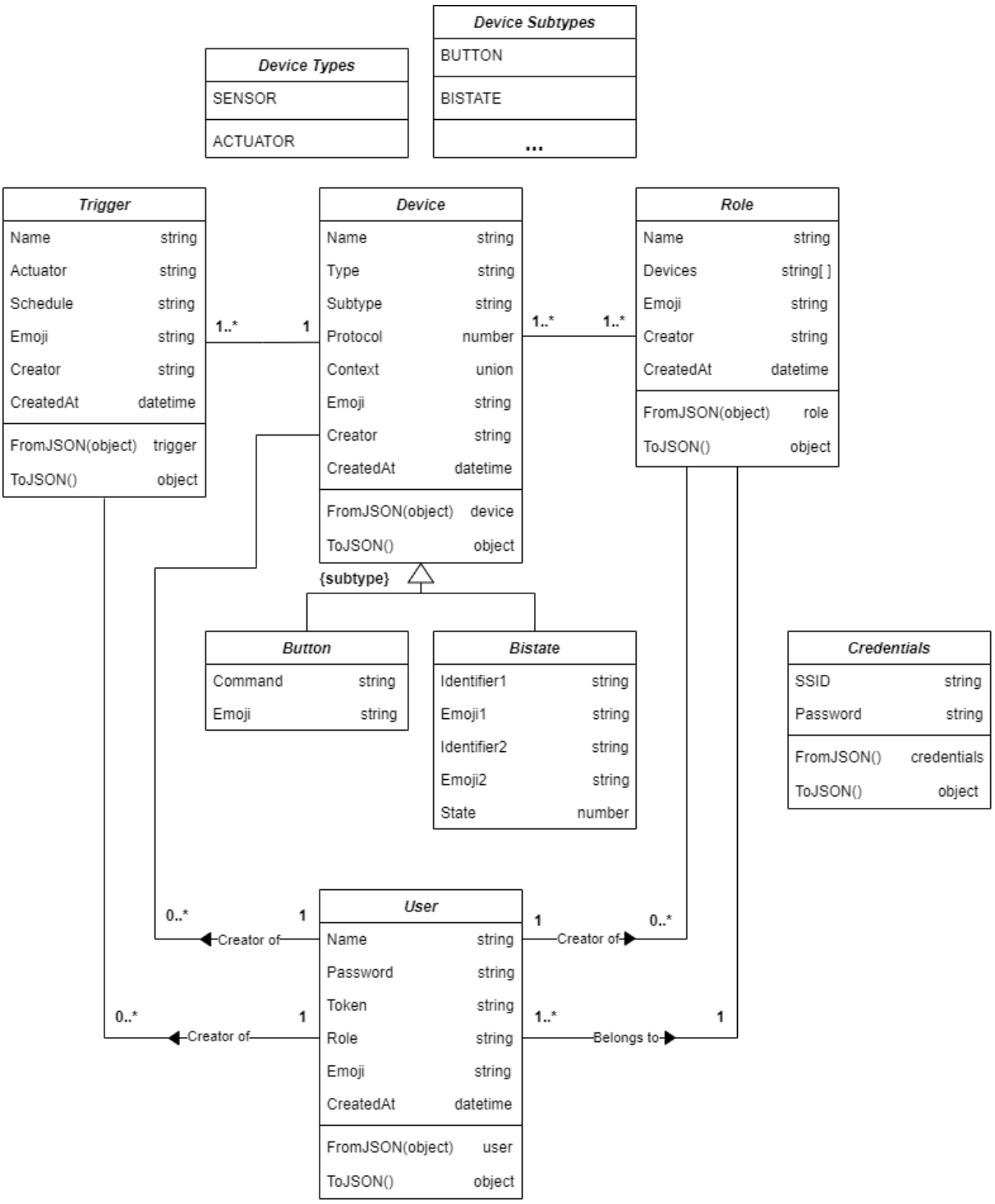


Figure 23: UML diagram of the backend models. *Source: Compiled by the author*.

The credentials object represents the stored user's home network SSID (name) and password and does not have relations to any other model. The password is stored encrypted to avoid data leak.

The user object represents a user of the system that interactuates through the HTTP REST API. As such, it includes a name, which identifies her/him, and a password, along with a token, which as explained later on is the authentication mechanism used through all the API to be able to authenticate the identity of the requesting user and perform access control. The password is stored as a sha256 hash to avoid data leak

Moreover, it also includes a role, to perform RBAC; an emoji, which identifies its picture (explained later on the *frontend* section); and the timestamp when it was created.

The user model has several relations with the other models:

a) Roles: a user belongs to a role and a role can include multiple users. Moreover, admin users can create and manage roles, and a role can only have one creator.

b) Devices: a user is the creator of one or more devices and a device can only have one creator.

c) Triggers: a user is the creator of one or more triggers and a trigger can only have one creator.

The device object represents a connected sensor or actuator. It is identified by a name and can be either a sensor or actuator type.

Moreover, a subtype attribute is included to further discern the different sensors and actuators, this is helpful because an alarm and a weather station both are sensors but its usage, displayal and communication needs are pretty different.

There can be as many subtypes as implemented, such as switches, gauges or sliders, but to keep it concise only two have been implemented which correspond to the used test subjects: open-close alarm, bistate; wireless switch, button; and wireless motorized blinds, button.

Depending on the subtype a device can have different contexts:

a) Button: includes the actuator's command data string and an emoji to display on the button in the dashboard.

b) Bistate: includes the information for the sensor's two states: the identifier data string and the emoji to display the corresponding state in the dashboard. The "state" attribute saves the last state known (received) by the system.

Following, the protocol attribute identifies the communication protocol implementation of the device, similar to a device major/minor driver number in POSIX[26]. A full explanation of the transmission and reception mechanism, protocols and packet implementations lies in the *firmware* section of this thesis.

Moreover, the object also includes an emoji for the resource itself, the creator user's name and the timestamp when it was created.

The trigger object represents a stored scheduled action on an actuator. It is identified by a name and it is related to the actionable device through the "actuator" attribute. A trigger can only fire an actuator but an actuator can be scheduled through multiple triggers.

The schedule itself is expressed using the cron syntax, as seen below in *Figure 24*. There are several standards for such syntax which include seconds or years, but it is not common for anyone to schedule an action on a device every nth-year and to avoid overload on the system the minimum unit of schedulement has been set to 1 minute. A more in-depth explanation of the implementation for such scheduler lies in the *firmware* section of this Thesis document.

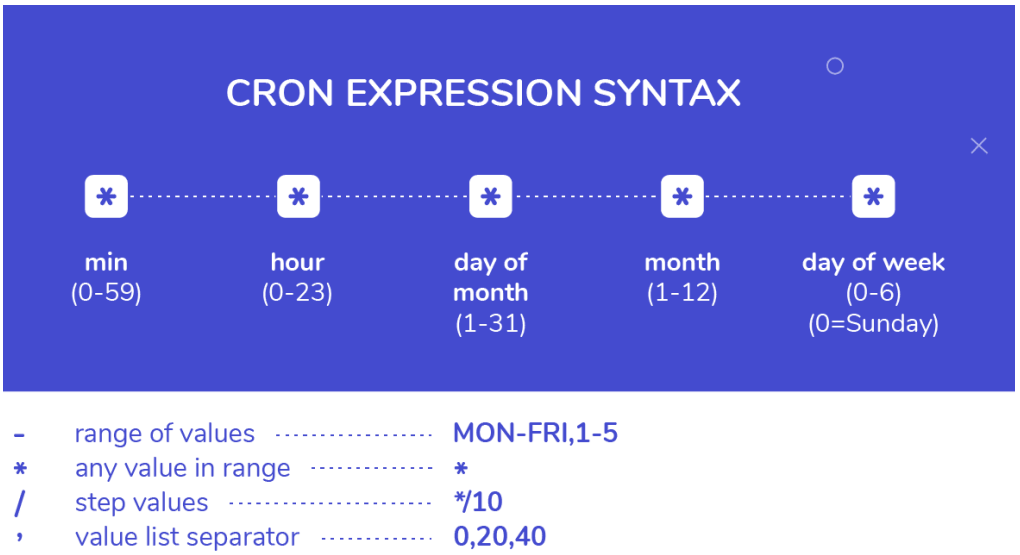


Figure 24: Cron expression syntax. *Source: Moty Michael* [44].

Moreover, the object also includes an emoji for the resource itself, the creator user's name and the timestamp when it was created.

The role object represents a collection of devices to which the belonging users can access to and is identified by a name. Through the "devices" attribute the role is related to such devices. A role can include multiple devices and a device can belong to multiple roles.

[26] Unix Portable Operating System Interface.

There are two default system roles, which cannot be updated nor deleted:

a) Admin: this role is automatically assigned to the first user created on the system. An administrator can access all devices no matter what other roles are created, and can promote users to roles or even to admins. Moreover, they can perform additional actions such as changing the connected network or factory resetting the system.

b) Guest: this role is automatically assigned to all users when they are registered. A guest cannot access any device until an administrator promotes the user to another role that includes devices. This is a security mechanism as users can be registered at any time without restriction.

The enforcing of role based access control on API resources is explained in more detail in the *firmware* section of this Thesis document. Moreover, the object also includes an emoji for the resource itself, the creator user's name and the timestamp when it was created.

In order to transfer object information to the database layer or through the HTTP REST API all models are serialized or deserialized from JSON[27], providing and standardized encoding for storing and serving resource data.

As this is an embedded system with limited resources in RAM, ROM and CPU, the proposed storage system for the data is a simple key-value database where the pairs are stored in the built-in flash memory (ROM) of the chip. Such database is also known as NVS within the ESP-IDF framework.

Moving on to other software components, the provisioner is in charge of providing Internet connectivity to the system through Wi-Fi or managing its softAP network if unable, so it can be in two modes:

a) Station: in this mode the device acts as a client (station) of an existing Wi-Fi network. On the initial provision, or when the user requests it, the provisioner scans for available Wi-Fi networks in the environment using the scanner component to which the user can select one to connect to providing its credentials.

   The minimum accepted authentication mode is WPA2[28] and the provisioner will negotiate with the network's DHCP[29] server the static IP that the user can also request.

   These credentials will be stored to automatically provision the system if the system reboots or the connection is lost. After 10 retries, if the system cannot connect to the target Wi-Fi network the provisioner switches automatically to softAP mode.

[27] JavaScript Object Notation.
[28] Wi-Fi Protected Access 2.
[29] Dynamic Host Configuration Protocol.

b) SoftAP: in this mode the device acts as the router or gateway of its own created Wi-Fi network. The device starts in softAP mode on the initial provision, as there are no stored credentials, or switches to it as a fallback when the network connection is down. The created network name is "Diana dot" and it is authenticated using WPA2. The credentials for the network are unique to each device and come with the packaging.

   As required in the *requirements* section the system is fully usable through the softAP connection, as the other software components of the system have been decoupled from the network connection.

   In order for the user not to be lost when she/he connects to the device's Wi-Fi network, as the user must know the device's IP, a DNS server has been implemented that answers all the name resolution requests with the device's IP, creating what is called a captive portal. If the user navigates to any address the DNS server will redirect her/him to the device's web dashboard. On smartphones this is actually done automatically when the user connects to the network as the operating system recognizes the existence of a captive portal.

A more detailed provisioning flowchart is available in the *firmware* section of this Thesis.

The other principal software component is the HTTP server. It is the application transport layer for the system's REST API, which exposes all of its resources to a frontend application and also allows it to manage them. The HTTP server functions separately from the TLS encryption layer, which is in charge of the TLS/SSL connection handshake whenever the server functions through HTTPS.

The server is separated in three main endpoint groups:

a) Frontend: this group catches the root and all not found requested paths and serves the frontend web application file instead.

   The frontend code is embedded on the flash memory of the microcontroller into a single minified and compressed HTML file using a FAT[30] filesystem mounted on the flash memory of the chip (ROM). The file is downloaded, interpreted and expanded automatically by the browser, therefore only requiring one single request to the microcontroller to download the full application. Moreover, HTTP cache control headers have been used to prevent browsers from requesting the application each time a user accesses it.

[30] File Allocation Table.

b) Authentication: this group catches both the registration and login paths. When a new user is registered the system saves it (the password is stored hashed through sha256) and generates a cryptographically random token which will serve as the authentication claim needed for requesting other API endpoints. That token will authorize the identity of the requesting user in the system in order to enforce role based access control on the resources.

   A new authentication token is generated every time the existing user successfully logs in with its name and password, therefore closing the previous opened session avoiding data leaks.

c) API: this group catches the users, devices, triggers, roles and system paths. System endpoints allow to retrieve the device's connected network and its time, manage system configuration options and factory reset the system which erases all content of the database.

   The other endpoints allow to retrieve, create, update or delete the different resources of the system depending on the used HTTP method: GET, POST, PUT or DELETE. Moreover the devices sub-group has an additional endpoint to manually fire an actuator. Every endpoint firstly performs user authentication and ensures RBAC.

Several detailed endpoint flowcharts are available in the *firmware* section of this Thesis.

Moving on to other important software parts, the chron component is in charge of maintaining the current system time correctly, critical to the trigger scheduler functionality.

It features three levels of system synchronization:

a) Built-in timer: the first layer of time synchronization is the high-resolution hardware timers of the microcontroller itself. A hardware timer will keep the counting of the current timestamp since the epoch[31] continuously, incrementing it every second. The other software components that require the current time will get it directly from this timer.

b) RTC module: the second layer of time synchronization is the external real-time clock module on the PCB. The RTC will be connected through an I2C interface and is the source of time for the first layer. Every 5 minutes the microcontroller will synchronize its time taking the current time from the RTC and updating its hardware timer. This layer is critical to ensure keeping the time correctly when the device cannot be connected to the Internet.

[31] Date and time from which a computer measures system time, generally Thursday 1, January 1970 00:00 UTC.

c) NTP connection: the third and final layer of time synchronization is the network time protocol connection that the microcontroller maintains at all times which is the source of time for the second layer. This layer is ignored when the device cannot reach the Internet. Every hour the microcontroller will synchronize the time of the RTC taking the current time from an NTP request and updating the RTC timer.

The user can select its current time zone by updating the system configuration.

It must be noticed that time adjustments from the RTC to the system are performed smoothly, by gradually reducing time error. However, if the difference between both clocks is greater than half an hour the time is corrected immediately, in order to not delay incoming events and to not trigger actuators so late that the user may no longer wants. Time adjustment is performed using the libc function "adjtime".

Another important component is the status, which is in charge of controlling the RGB LED. The controller functions as a queue of requests to change the RGB LED color. Statuses are executed concurrently, that means, a request can hijack the current status and then switch again to the previous status if that status was configured to be "looping". The show time of the statuses can be either bounded or unbounded and can also blink by toggling the LED.
Other software components can request asynchronously several different statuses:

a) Idle: looping status with indigo color. It is triggered when the device has correctly connected to the target Wi-Fi network or has started its softAP network. Never ceases.

b) Transmitted: flashing status with green color. It is triggered when a packet has been transmitted through the transmitting module.

c) Received: flashing status with white color. It is triggered when a packet has been received through the receiving module.

d) Provisioning: blinking status with yellow color. It is triggered when the device is trying to connect to a target Wi-Fi network or it is setting up its softAP network.

e) Error: blinking status with red color. It is triggered when the device cannot be connected to the target Wi-Fi network or has lost connection. Ceases if the device is able to reconnect to the network or when the fallback softAP network is started after 10 retries.

f) Off: looping status with no color. It allows to switch off the LED. It is currently never triggered.

A more detailed status controller flowchart is available in the *firmware* section of this Thesis.

The logger component is in charge of logging debug output messages through the JTAG interface that can be monitored through a command line. It allows to track the execution of the firmware in real time and trace any possible error while developing the firmware. Once the firmware is fully completed, the logger can be switched off to avoid wasting unnecessary resources. The output levels are: debug, info, warn and error.

The over-the-air (OTA) component is in charge of updating the backend firmware by downloading an encrypted and signed firmware binary over HTTP from a host over the Internet or a host of the network. The HTTP client functions separately from the TLS encryption layer, which is in charge of the TLS/SSL connection handshake whenever the client functions through HTTPS.

After the full binary has been downloaded and verified, the microcontroller flashes itself and performs a hardware reboot to boot from the new firmware. The binary is never decrypted as the chip supports flash encryption, see the following.

All of the previously mentioned backend firmware code, the database and the frontend are stored in the flash memory of the microcontroller (ROM). In order to ensure that the system is highly secure two security layers have been implemented:

a) Secure boot: this mechanism protects the device from running any unauthorized code by checking that the bootloader and the firmware binary is cryptographically signed by the original developer using RSA-PSS[32].

   If any part of the code has been maliciously changed or injected through the JTAG interface the device will abort the booting process and brick itself until a new correct signed binary is flashed.

   The microcontroller uses the RSA public key of the developer, burnt before to one of its e-fuses[33], to verify the software integrity.

b) Flash encryption: this mechanism encrypts the contents of the chip flash memory using the AES512-XTS algorithm.

   Firmware is flashed as plaintext, and then the data is encrypted in place on the first boot. As a result, a physical readout of flash is not sufficient to recover its contents avoiding the dump of the internal bootloader and firmware code, database and frontend code.

   The microcontroller uses a random auto generated RSA private key, and burns it to one of its e-fuses, every time a new firmware binary is flashed.

[32] Rivest–Shamir–Adleman Probabilistic Signature Scheme.
[33] Electronic Fuses which allow for the dynamic real-time reprogramming of chips once or multiple times.

Frontend

The following figures depict the simplified conceptual diagrams for the pages and sections of the frontend web application of the firmware. In this case, a diagram for the code is not included because it does not present any notable complexities and it does not constitute a main part of this Thesis. The frontend has been implemented as an SPA[34] using the JavaScript framework *Svelte* [45] through the TypeScript flavor. The framework has been selected because of its simplicity and low memory footprint and TypeScript because of its static type safety.

It is worth mentioning that, as seen in the previous *Figure 22*, the frontend code is embedded on the flash memory of the microcontroller, which means a size revision mechanism must be imposed. In order to reduce its size, the code has been both minified (code shortening) and then compressed using Gzip[35], decreasing by more than a factor of 10 its proportions.

Moreover, in order to reduce the amount of requests to the HTTP server on the microcontroller, all frontend resources: HTML, CSS and JavaScript; pages, sections and components have been bundled into a single HTML file which is downloaded, interpreted and expanded automatically by the browser, therefore only requiring one single request to the microcontroller to download the full application. Even more, HTTP cache control headers have been used to prevent browsers from requesting the application each time a user accesses it.

The following *Figure 25* illustrates a simplified diagram of the dashboard page. It is the main page of the web application where the users can navigate to the different sections: devices, triggers, users and roles; the system configuration pages and see the status bar.

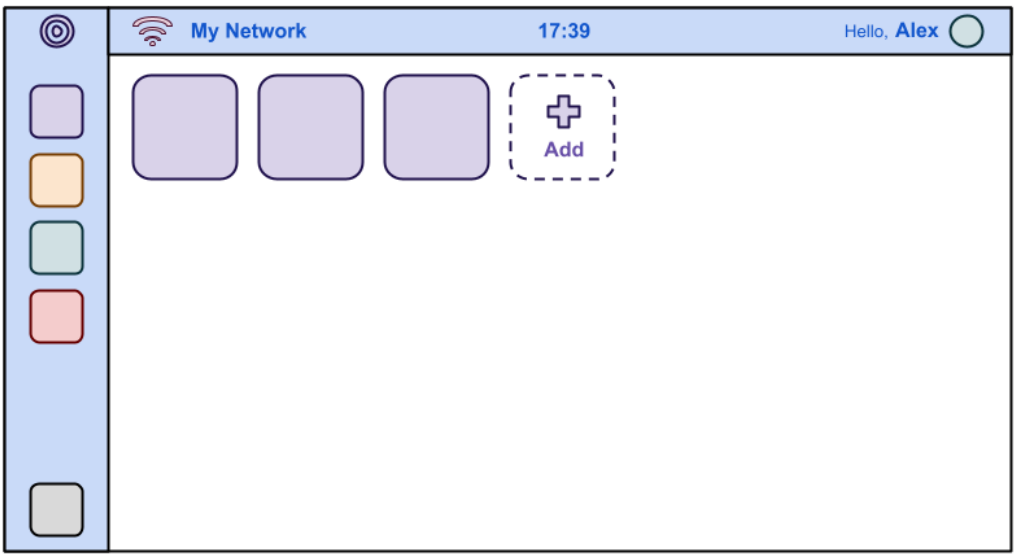


Figure 25: Conceptual diagram of the frontend dashboard page.
Source: Compiled by the author.

[34] Single Page Application.
[35] GNU Zip compressing algorithm.

Starting at the status bar, it features 3 main pieces of information: the current network connected to or the softAP network if not, the current time from the device and the current authenticated user.

On the left there is the sidebar navigation that allows the user to switch to other tabs. These sections will be displayed inside the dashboard layout for easy access to both the status and side bars. From the top, the tabs are devices, triggers, users, roles and system information page.

Sections will display a grid of components known as "cards". The information included in those cards varies depending on the type as seen in the following simplified diagram *Figure 26*.

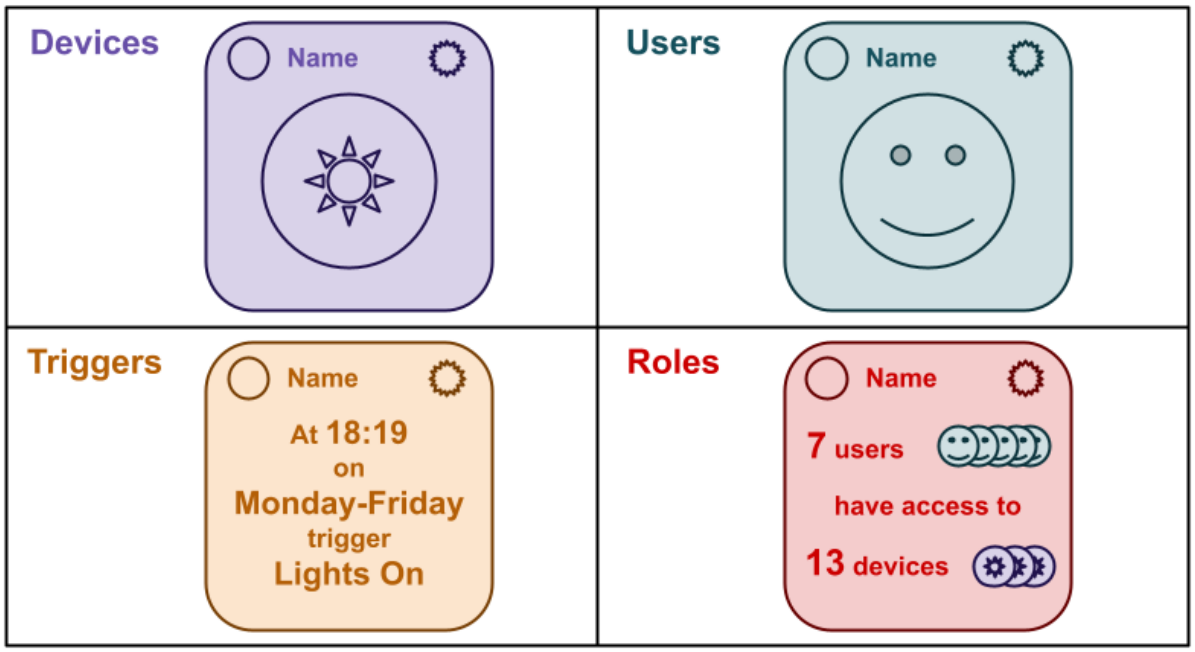


Figure 26: Conceptual diagram of the frontend devices, triggers, users and roles sections.
Source: Compiled by the author.

On all cards the header part is similar, including the emoji (picture) and name of the resource. On the left side there is a settings button that opens up a modal including its full information, an update and delete button. To create resources there is an add button at the end of each grid.

It must be noticed that emojis are used instead of pictures or icons. The reason is that it allows to both decrease size and complexity of both backend and frontend systems. Emojis are part of the UTF-8 character set used by all modern browsers, and thus do not require a separate storage.

The body of the device card depends on its type and subtype. For actuator buttons a single emoji is displayed that serves as the button. For bistate sensors two different emojis are displayed depending on the current state of the sensor.
The body of the trigger card displays the schedule specification in human readable text, emphasizing the actual dates and hours.

The body of the user card is a big version of the user's emoji along with its role underneath.

Finally, the body of the role card contains the number of users and their emojis which have access to the different devices the role includes. The number of devices and their emojis are also included

Going into the system page the user can see several information such as the chip model, revision, MAC[36] address, softAP network name and password; firmware and SDK[37] version; resource and storage usage.

Here the user can also scan the available Wi-Fi networks and change the current connected network. Moreover, a reset system button is included which erases all contents of the embedded database.

The following *Figure 27* illustrates the simplified diagram for such page.

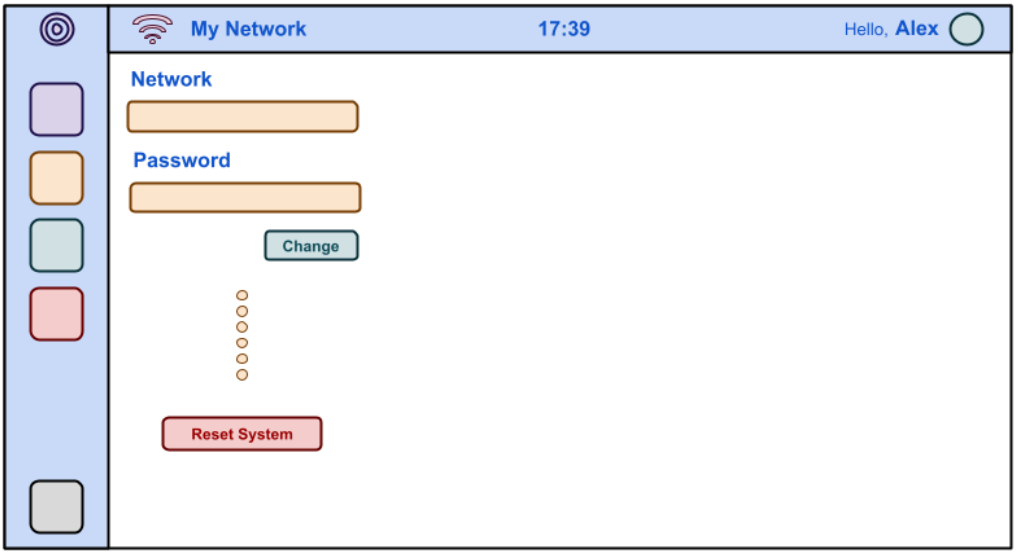


Figure 27: Conceptual diagram of the frontend system page.
Source: Compiled by the author.

In order to authenticate and create new users with the previously explained mechanism in the conceptual diagram for the *backend firmware*, an authentication page is needed. The

[36] Media Access Control.
[37] Software Development Kit also known as toolchain.

following *Figure 28* depicts the login or register page. The interface is the common authentication form with an input box for the email, for the password and a "remember me" checkbox that will save the obtained authentication token to the browser's local storage. In order to register a second password-validation input box will be shown when the user clicks on the "or register" button.

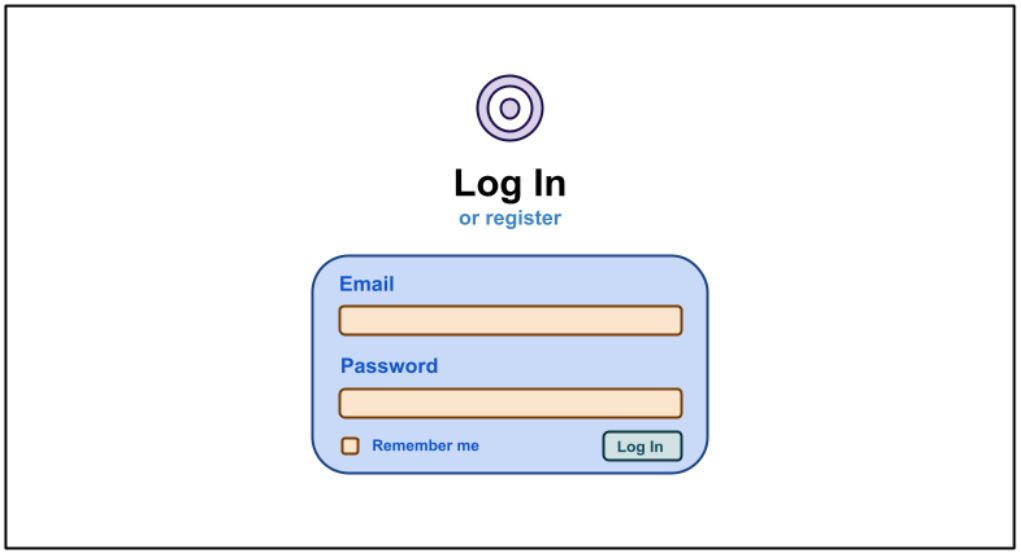


Figure 28: Conceptual diagram of the frontend authentication page.
Source: Compiled by the author.

Finally an onboarding page, *Figure 29*, has been included to aid the user when the system is new (or has been reset), including a register and select Wi-Fi forms for a fast system setup. The user is automatically redirected to this page if there are no users in the system when she/he connects to the softAP network thanks to the built-in captive portal.

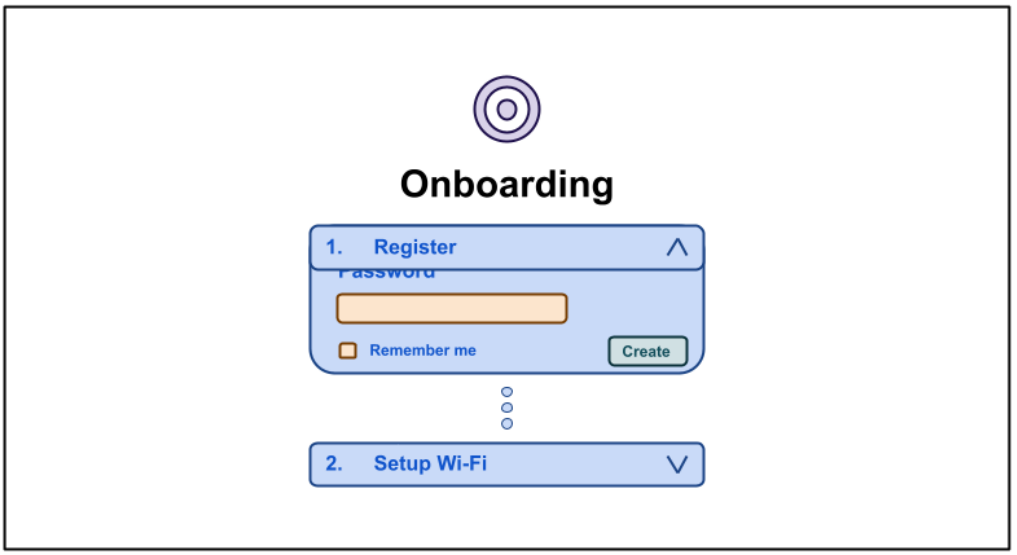


Figure 29: Conceptual diagram of the frontend onboarding page.
Source: Compiled by the author.

9. Hardware Components

The electronic device is composed of several hardware components. The following *Table 9* contains the complete bill of materials (BOM) along with the unitary and wholesale cost of each component. The table also includes the operating and absolute voltage in volts; and the absolute maximum and drawn current in milliamperes, that is the power consumption. There are components to which the operating voltage and drawn current is not applicable. The components have been bought through the electronics distributor *Farnell* [46]. In a chip-shortage era it was difficult to get together all the required parts.

NAME	UNITS	COST		VOLTAGE		CURRENT	
		x1	x1000	OPERATING	MAX	DRAWN	MAX
ESP32-S3-WROOM-2-N32R8V	1	7.34	7.34	3.3	3.6	355.0	1,500.0
QAM-RX10-433	1	4.38	2.46	5.0	5.5	6.0	600.0
QAM-TX2-433	1	5.51	2.63	5.0	12.0	15.0	850.0
W3127	2	1.90	0.88		46.0		920.0
DS3231SN	1	4.06	2.80	3.3	6.0	0.7	1,000.0
CLX6F-FKC	1	0.34	0.26	5.0	5.0	35.0	100.0
BC847B	4	0.16	0.06		45.0	10.0	200.0
USB4085	1	0.75	0.62		20.0		3,000.0
USBLC6-2SC6	1	0.65	0.29		15,000.0	0.15	2,500.0
LDL1117S33R	1	1.38	0.79		18.0	0.5	1,200.0
MP008293	1	0.11	0.05	5.0	5.0	20.0	75.0
MF-MSMF075/24	1	0.32	0.30		24.0		1,500.0
1N5819HW-7-F	1	0.53	0.10		40.0	0.05	1,500.0
KMR421G-LFS	2	0.72	0.72		32.0		50.0
RESISTOR 0Ω	3	0.01	0.01		300.0		3,500.0
RESISTOR 22Ω	2	0.01	0.01		300.0		3,500.0
RESISTOR 330Ω	3	0.01	0.01		300.0		3,500.0
RESISTOR 1kΩ	4	0.01	0.01		300.0		3,500.0
RESISTOR 4.7kΩ	2	0.01	0.01		300.0		3,500.0
RESISTOR 5.1kΩ	3	0.01	0.01		300.0		3,500.0
RESISTOR 10kΩ	2	0.01	0.01		300.0		3,500.0
CAPACITOR 0.1µF	5	0.01	0.01		100.0		25,000.0
CAPACITOR 1µF	2	0.01	0.01		25.0		6,250.0
CAPACITOR 4.7µF	1	0.02	0.01		10.0		2,500.0
CAPACITOR 10µF	3	0.3	0.01		16.0		4,000.0
CAPACITOR 22µF	1	0.05	0.01		6.3		1,575.0
INDUCTOR 6.8nH	2	0.31	0.9		49.0		800.0
INDUCTOR 12nH	2	0.63	0.30		42.0		700.0

Table 9: Bill of materials, electrical characteristics and absolute maximum ratings. Cost in euros, voltage in volts and current in milliamperes. *Source: Compiled by the author*.

9.1. Microcontroller

The microcontroller is the 3.3V SoC[38] *ESP32-S3-WROOM-2-N32R8V* [47] from Espressif. Specifically it belongs to the ESP32-S3 family and features a dual core 32-bit LX7 chip up to 240 MHz which comes with integrated WiFi and Bluetooth connectivity. The N32R8V variant comes with 32 MB of flash memory, 512 KB of SRAM and 8 MB of PSRAM. Moreover, it includes several peripherals and has 33 usable GPIOs.

Figure 30: ESP32-S3-WROOM-2-N32R8V.
Source: Compiled by the author.

It was selected because:

a) Wi-Fi capabilities: allows to connect the device to the Internet or creating its own softAP network.

b) Dual core: allows not to block Wi-Fi/HTTP communication while the device is receiving or transmitting commands.

c) Resources: the CPU, RAM and ROM onboard are more than sufficient for the firmware, database and the embedded frontend.

d) Peripherals: the available and enough GPIOs all have PWM and interrupt capabilities and the required I2C interface is also provided.

e) USB JTAG controller: allows to program (flash) and power the microcontroller through a single USB-C connector.

f) System on a Chip: the microcontroller comes already with flash memory and RAM integrated onto a single package and has a PCB antenna onboard, therefore decreasing drastically the complexity of the electrical schematic and the PCB itself.

g) Development environment: the ESP32-X family of chips has a big community built around, and with the framework given by the vendor every feature thought was feasible, even the security ones.

[38] System on a Chip.

9.2. Receiver

The 433MHz receiver is the *QAM-RX10-433* [48] from Quasar UK. Comes as a compact module with a small footprint. It can receive AM signals in the 433MHz frequency and has an enhanced noise-immunity internal PLL[39] which constantly adjusts to match the frequency of the incoming signal. The module works at 5V and outputs the received data directly through a TTL pin. It comes with an antenna but another type was used instead.

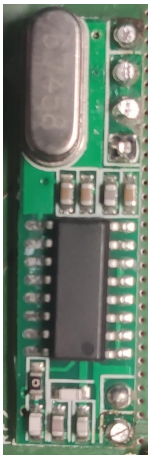


Figure 31: QAM-RX10-433.
Source: Compiled by the author.

It was selected because it works at 433MHz frequency, offers a good reception range (+30m) and a good throughput (10kb/s), is a small form factor, has a low power consumption (6mA) and works at 5V. Moreover, as it is a module the complexity of the electrical schematic and the PCB itself decreases drastically.

9.3. Transmitter

The 433MHz transmitter is the *QAM-TX2-433* [49] from Quasar UK. Comes as a compact module with a small footprint. It can transmit signals using ASK modulation in the 433MHz frequency. The module operates from 3V up to 12V, higher the voltage higher the transmission range. Data can be sent to the transmitter directly via a single TTL pin. It does not come with an antenna.

Figure 32: QAM-TX2-433.
Source: Compiled by the author.

It was selected because it works at 433MHz frequency, offers a good transmission range (+50m) and a good throughput (3kb/s), is a small form factor, has a low power consumption (15mA) and works at 5V. Moreover, as it is a module the complexity of the electrical schematic and the PCB itself decreases drastically.

[39] Phase-Locked Loop.

9.4. Antennas

The antennas are the *W3127* [50] from Pulse Electronics. Comes as a compact through-hole horizontal helical antenna. It is tuned to work in the 433MHz band. It has a single end pin where data can be sent to or received from.

Figure 33: W3127.
Source: Compiled by the author.

It was selected mainly because it works at the 433MHz band and can be directly mounted onto the PCB in a horizontal manner securely, therefore the casing can maintain a low profile.

9.5. Real time clock

The real time clock is the *DS3231SN* [51] from Maxim Integrated. Comes in a small integrated circuit and it is extremely accurate as it includes a temperature compensated crystal oscillator. The RTC can be controlled through I2C, has several timekeeping functions and interrupt alarms. The IC operates at 3.3V.

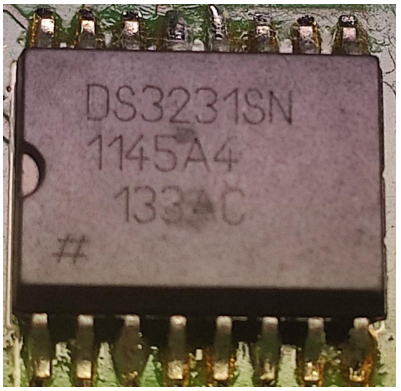


Figure 34: DS3231SN.
Source: Compiled by the author.

It was selected because it works at 3.3V, is extremely accurate (±2ppm) and the required operating circuit is very simple, thus drastically decreasing the complexity of the electrical schematic and the PCB itself.

9.6. RGB LED

The RGB LED is the *CLX6F-FKC* [52] from Cree. Comes as a not-so small 3 in 1 SMD LED with a semi translucent layer on top to diffuse the light. It is not common anode or common cathode, to control the colors of the LED PWM signals have to be sent to each pin channel separately. As common LEDs it operates at 5V.

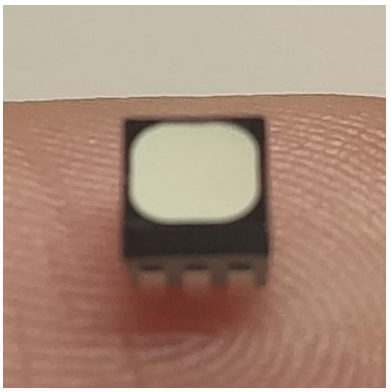


Figure 35: CLX6F-FKC.
Source: Compiled by the author.

It was selected mainly because of its big size and the fact that it is already diffused by default.

The microcontroller works at 3.3V and its GPIOs cannot supply the sufficient amount of current needed to fully brighten the LED. NPN transistors are needed to step up the current given to the LED pins, using directly the PCB 5V power input, while maintaining the PWM signal to be able to change colors.

The following *Figure 36* shows the *BC847B* [53] NPN transistors from Multicomp Pro that were used.

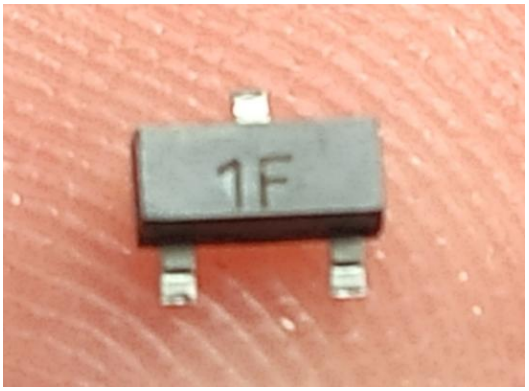


Figure 36: BC847B. *Source: Compiled by the author*.

It was selected mainly because its NPN, its maximum collector current (200mA) is sufficient for the maximum current drawn from the RGB LED (100mA) and the base current needed to saturate the transistor is low (10mA).

9.7. USB-C connector

The USB-C receptacle is the *USB4085* [54] from GCT. It comes as a trough-hole horizontal low profile connector. It includes the common pins of USB 2.0 and a few other pins required in the USB-C specification. Those pins can be configured to let the connected end know that the device works at full-speed. SMD was preferred but there were not available options.

Figure 37: USB4085.
Source: Compiled by the author.

It was selected mainly because it is USB-C 2.0, can withstand (3A) more than the maximum current drawn by all the system (750mA) at all times and has a low profile, therefore the casing can maintain it too.

9.8. Voltage regulator

The voltage regulator is the *LDL1117S33R* [55] from STMicroelectronics. It is a small linear regulator which can reduce the input voltage from 18V~3.7V down to 3.3V. It comes with a small heatsink where all of the excess power is dissipated as heat. Includes a thermal protection mechanism that switches it off when it is overheated to avoid damages.

Figure 38: LDL1117S33R.
Source: Compiled by the author.

It was selected mainly because it can lower the voltage from 5V to 3.3V (for the components that require it) and its guaranteed output current (1.2A) is more than sufficient to supply the maximum current drawn by all the system (750mA) at all times. Moreover, its quiescent current (power consumption) is low (0.5mA).

In order for the user to assert that the device receives power and is switched on, a red LED is included in the PCB, although it is dimmed to not waste energy. The following *Figure 39* depicts the *MP008293* [56] red LED from Multicomp Pro used.

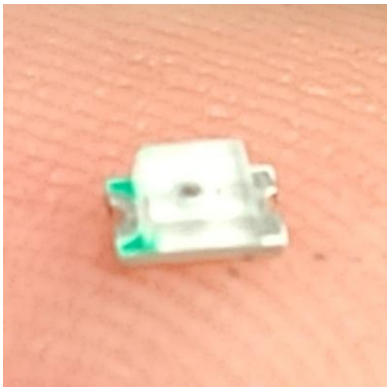


Figure 39: MP008293. *Source: Compiled by the author*.

It was selected because it is SMD and has a competitive price.

9.9. Circuit protectors

Over-current

The polyfuse for the over-current protection circuit is the *MF-MSMF075/24* [57] from Bourns. It is a small resettable fuse that when the current flowing through it exceeds a certain threshold, it "trips" and opens the circuit, interrupting the flow of current. The polyfuse is automatically resetted when the input current decreases down to the “hold” threshold.

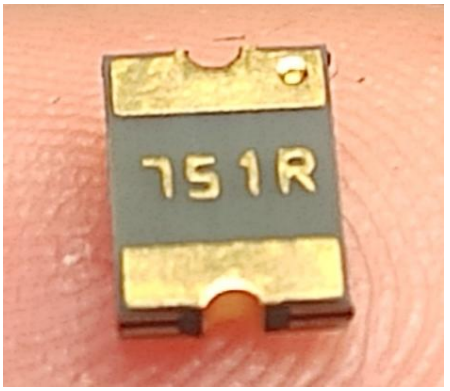


Figure 40: MF-MSMF075/24.
Source: Compiled by the author.

It was selected mainly because its trip current (1.5A) corresponds to the absolute maximum current that the system can withstand, consequently the interval to its resettable hold current (750mA) is secure for the system.

Over-voltage and reverse bias

The schottky rectifying diode for the over-voltage and reverse bias protection circuit is the *1N5819HW-7-F* [58] from Diodes Incorporated. It blocks the current flowing to the other direction and will be destroyed when inputted high voltage. Schottky diodes have a lower forward voltage drop than regular diodes.

Figure 41: 1N5819HW-7-F.
Source: Compiled by the author.

It was mainly selected because its maximum rectified output current (1.5A) corresponds to the absolute maximum current that the system can withstand and its maximum forward voltage drop (0.45V) is as small as available, thus not lowering the overall operating voltage of the system too much.

Electrostatic discharge

The integrated circuit for the ESD protection circuit is the *USBLC6-2SC6* [59] from STMicroelectronics. Inside the small package there is an array of diodes that protects the data and power lines from ESD. It has very low capacitance between lines to GND for optimized data integrity and speed and is USB 2.0 compliant.

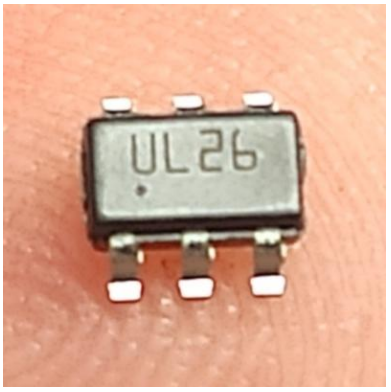


Figure 42: USBLC6-2SC6.
Source: Compiled by the author.

It was selected because it supports USB 2.0 ports up to high-speed (the device is full-speed), the maximum ESD that it withstands (15kV) is sufficient for a normal air spike (8kV) and includes protection for the USB power line.

9.10. Miscellany

Push buttons

The push buttons are the *KMR421G-LFS* [60] from CKSwitches. Come as surface mounted tactile buttons in a small footprint. The button comes with 4 pins being the two on each side the same.

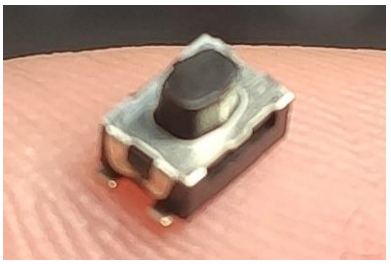


Figure 43: KMR421G-LFS.
Source: Compiled by the author.

They were selected because they are small SMD, maintaining a low profile on the casing.

Resistors

All the resistors used in the design are from *Multicomp Pro* [61]. Different resistors are needed for the required circuits of each other hardware component seen before.

Figure 44: Resistor sample.
Source: Compiled by the author.

Capacitors

All the capacitors used in the design are from *TDK* [62]. Different capacitors are needed for the required circuits of each other hardware component seen before.

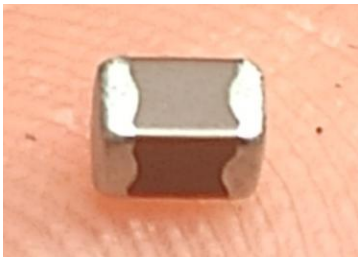


Figure 45: Capacitor sample.
Source: Compiled by the author.

Inductors

All the inductors used in the design are from *EPCOS* [63]. Inductors were needed for the tuning and matching circuits of both antennas.

Figure 46: Inductor sample.
Source: Compiled by the author.

All the resistors, capacitors and inductors were selected because they are 0805 SMD, that measure allows to decrease the overall PCB size.

10. Schematic

10.1. Design

Microcontroller

The following *Figure 47* illustrates the microcontroller part of the electrical schematic. The used symbol was given by the vendor.

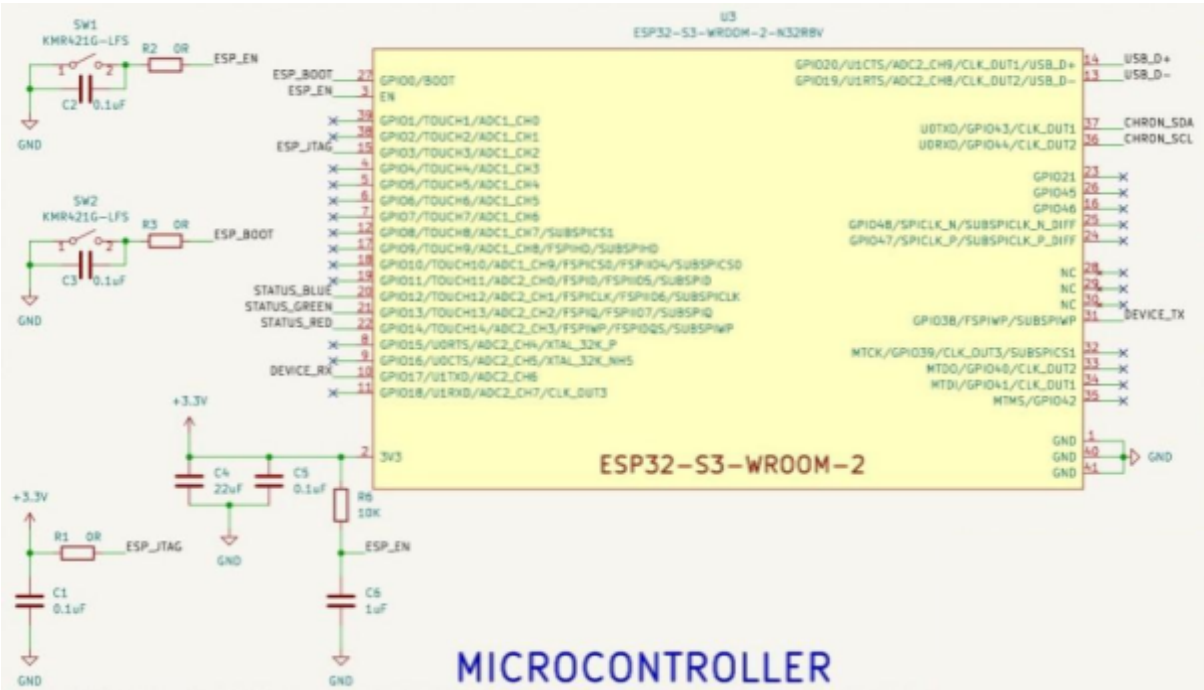


Figure 47: Microcontroller schematic part. *Source: Compiled by the author*.

Starting with the selected GPIO pins for controlling all other hardware components, the following *Figure 48* lists all the pin functions and features of the ESP32-S3-WROOM-2 series chip. The pins have been mainly selected taking into account their position on the PCB, as seen in the *PCB conceptual diagram*, to decrease the amount of connections crossing and making the routing less complex. In the following *PCB* section there is a diagram of the chip's pin layout.

Although not shown in the figure, all GPIO pins are interrupt-capable. The selected pin for the receiver data interrupt is the GPIO17. As the system uses only JTAG for flashing, monitoring and debugging, taking over the serial UART1 transmission line on that pin is all right.

For the transmitter input data the selected pin is the GPIO38. The RGB LED feature listed below refers to the built-in LED of the development board from which the list has been taken off. The other related functions on that pin are not applicable for this usage.

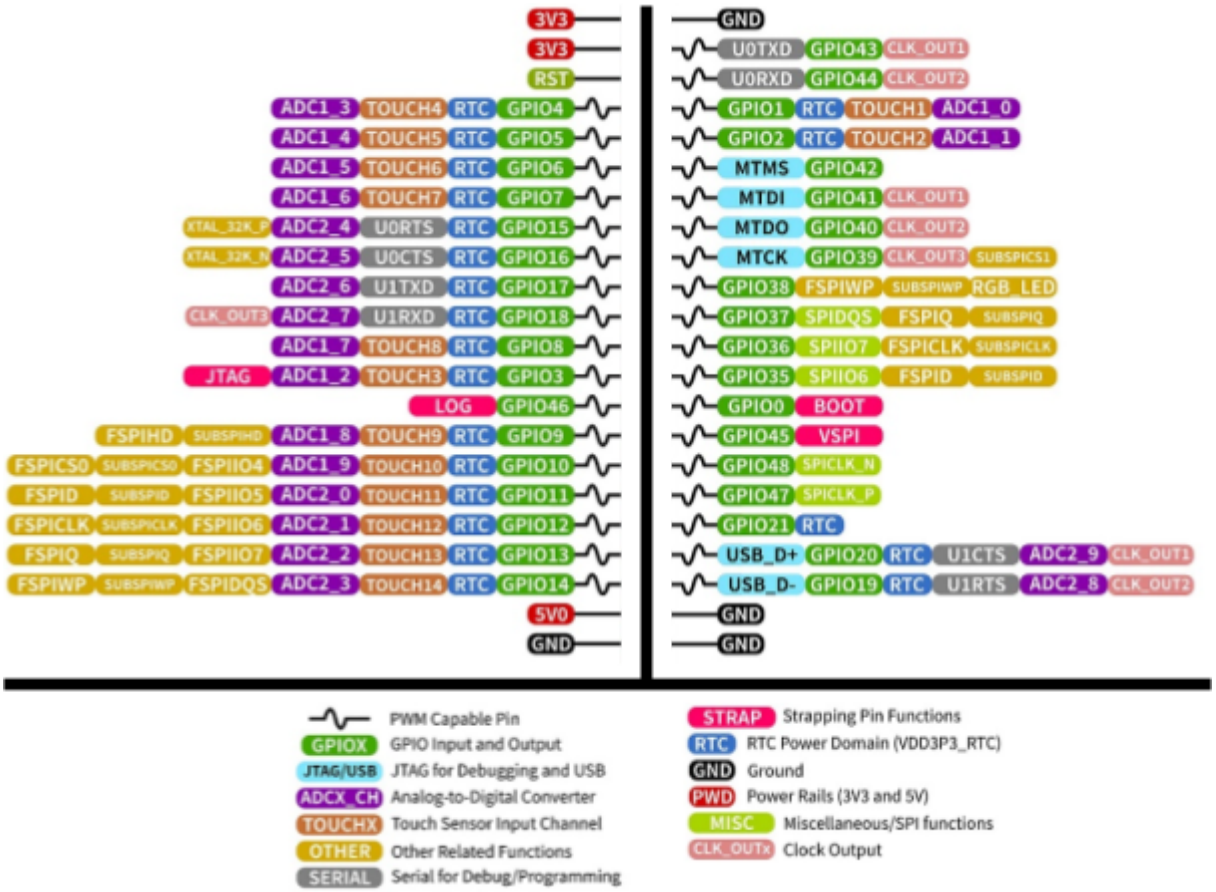


Figure 48: ESP32-S3-WROOM-2-N32R8V pin list. *Source: Espressif* [64].

The I2C interface can also be used on any GPIO pin, the underlying hardware accelerator will take care of it. The selected pins for the RTC SDA[40] and SCL[41] lines are the GPIO43 and GPIO44 respectively. To keep the relative functionality close together they were selected contiguously. Again, as the system uses only JTAG for flashing, monitoring and debugging, taking over the serial UART0 transmission lines on those pins is all right. The clock output function is not applicable for this usage.

As seen in the pin feature list all GPIOs are also PWM-capable. The pins selected for the red, green and blue channels of the RGB LED are the GPIO14, GPIO13 and GPIO12 respectively. To keep the relative functionality close together they were selected contiguously. All the other functions in those pins are not applicable for this usage.

Finally, the pins for the chip's built-in JTAG USB controller D- and D+ are the GPIO19 and GPIO20 respectively.As they are directly tied to the underlying hardware port the USB data line can only be in those pins. The other related functionality in those pins are not applicable for this usage.

[40] Serial Data.
[41] Serial Clock.

Thanks to the microcontroller model being specifically selected because of its SoC packaging, the required operating circuit for the chip is almost nonexistent. However, following the *ESP32-S3 series hardware design guidelines* [65] the chip should be activated after the power rails have stabilized, following the power on or system reset sequence shown in the following *Figure 49*.

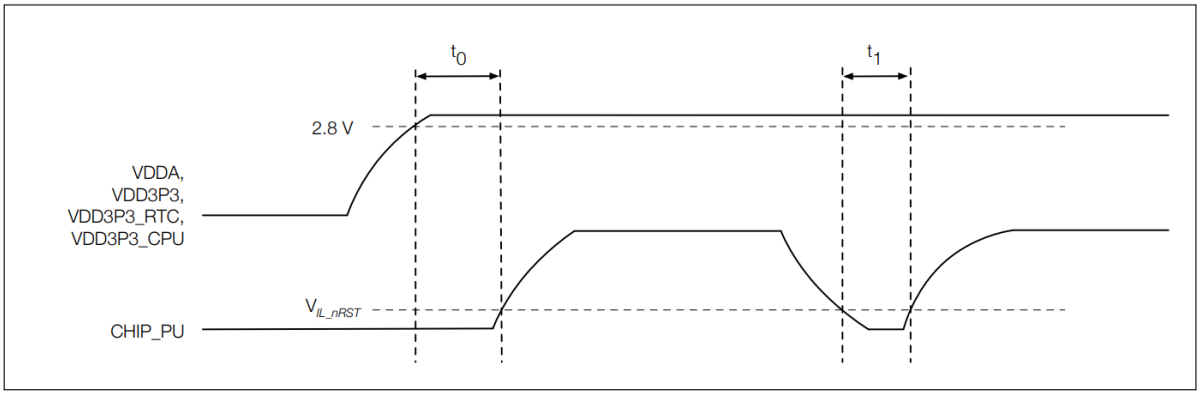


Figure 49: ESP32-S3 series power on sequence and system reset. "t_0" is the time (50µs) between bringing up the power rails of VDDA, VDD3P3, VDD3P3_RTC, VDD3P3_CPU and activating CHIP_PU. "t_1" is the duration (50µs) of CHIP_PU signal level $< V_{IL_nRST}$ to reset the chip. *Source: Espressif* [65].

As advised by the vendor, adding an RC delay circuit at the CHIP_PU (EN) pin ensures that stable power is supplied to the chip during power-up. The recommended setting for such RC delay circuit is a 10kΩ resistor (R6) and a 1µF capacitor (C6). Moreover, the 3.3V power input line has been furtherly stabilized by adding 22µF and 0.1µF capacitors (C4, 5).

Another important aspect to take into account when designing the microcontroller schematic is the configuration of its strapping pins. Those pins control the behavior and mode of operation of the chip on startup. The following *Table 10* summarizes the selected configuration of each strapping pin and its default value. To select one or the other configuration a logic "0" (GND) or a logic "1" (+3.3V) must be inputted to the corresponding pin before and through the chip's power on sequence.

The SoC packaging already comes with default values for the majority of such pins with the form of pull-up or down resistors. However, special attention has to be put on the JTAG signal selection pin, because it is floating by default.

The pin controls whether the chip will use the specific pins (GPIO39~42) for the JTAG interface or the USB controller pins (GPIO19, 20). As the JTAG will be used for flashing, monitoring and debugging through the USB-C connector, +3.3V has to be inputted to such pin. Moreover, it has been furtherly stabilized by adding a 0.1µF capacitor (C1).

SPI VOLTAGE			
PIN	**DEFAULT**	**3.3V**	**1.8V**
GPIO45	PULL-UP (N32R8V variant)	0	1
BOOTING MODE			
PIN	**DEFAULT**	**FIRMWARE**	**DOWNLOAD**
GPIO0	PULL-UP	1	0
GPIO46	PULL-DOWN	X	0
DEFAULT MESSAGES			
PIN	**DEFAULT**	**ENABLED**	**DISABLED**
GPIO46	PULL-DOWN	eFUSE	eFUSE
JTAG SIGNAL			
PIN	**DEFAULT**	**ON-CHIP PINS**	**USB CONTROLLER**
GPIO3	FLOATING	0	1

Table 10: ESP32-S3-WROOM-2-N32R8V strapping pins. *Source: Espressif* [47].

The circuit for the two push buttons, used to reset the system and to switch the booting mode on demand, are very simple. The only addition to them are two 0.1µF capacitors (C2, 3) to stabilize its values when pressed (debouncing), avoiding random reboots or mode switches.

Finally, the reason for the 0Ω resistors (R1, 2, 3) is because of the limited time available and inexperience. If any of the added circuits were to harm the microcontroller, it could be disabled by unsoldering the 0Ω resistor instead of having to cut the trace or remanufacturing an entire new PCB version again.

Receiver

The following *Figure 50* illustrates the 433MHz receiver part of the electrical schematic. The used symbol was designed from the data included in the *QAM-RX10-433 datasheet* [48].

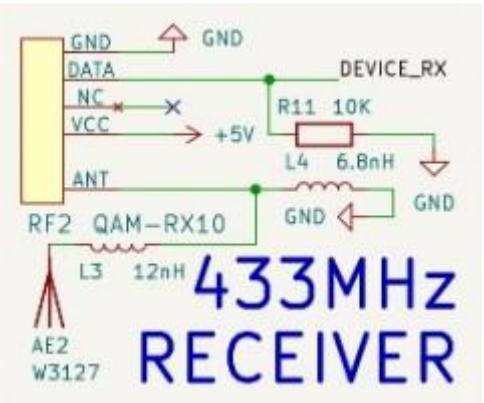


Figure 50: 433MHz receiver schematic part. *Source: Compiled by the author*.

Although in the component's pin list from its datasheet, seen in the *Figure 51*, there are two data output pins, in reality, as tested later on in the *testing* section, the receiver has only one real data output pin while the other one is not connected.

Pin	Description
1	Supply Voltage
2, 3	Data Output
4	Ground

Figure 51: QAM-RX10-433 pin list. *Source: Quasar UK* [48].

As said before and as seen in the schematic part, the receiver data pin, which works at 5V (TTL), is connected directly to a GPIO interrupt pin of the microcontroller because its GPIO withdands 5V. However, the chip works at 3.3V (LVTTL). As seen in the following *Figure 52*, the voltage input and output values marking high and low levels of both TTL and LVTTL overlap, thus making the two compatible without any extra circuitry needed.

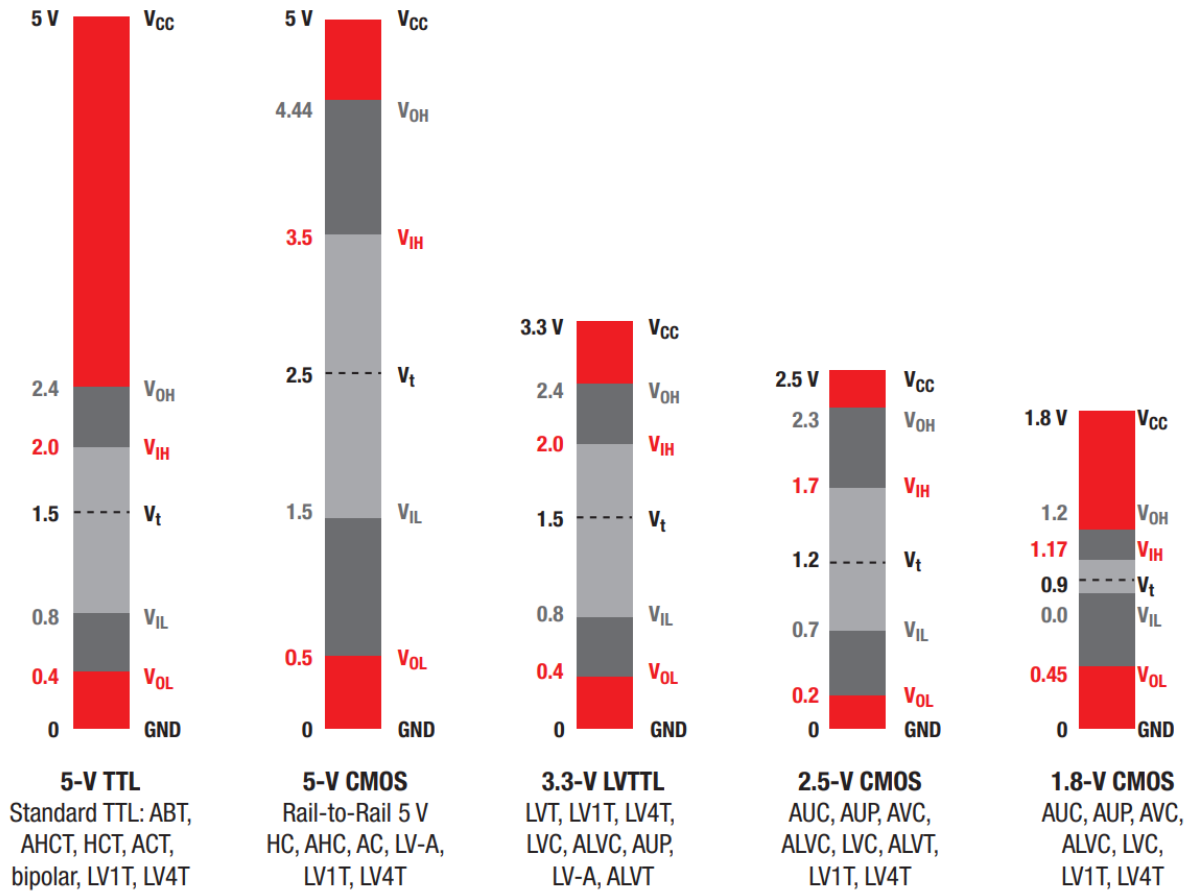


Figure 52: Logic level standards diagram. *Source: Texas Instruments* [66].

A pull-down resistor (R11) is included in order to reduce fake noise when the receiver did not receive any data.

Following the *W3127 antenna hardware design guidelines* [50], a matching and tuning circuit should be included in the design in order to calibrate the antenna. This type of circuit is usually composed of resistors, capacitors or inductors, and the exact values highly depend on the environment around the circuit. To properly design such circuits several expensive tools, like an VNA[42], are needed along with the complex and exhaustive knowledge of antenna design. As I lack both, the circuit used is taken out of the application example of the antenna's datasheet. It is composed of a 12nH inductor (L3) in series and a 6.8nH inductor (L4) in parallel.

Transmitter

The following *Figure 53* illustrates the 433MHz transmitter part of the electrical schematic. The used symbol was designed from the data included in the *QAM-TX2-433 datasheet* [49].

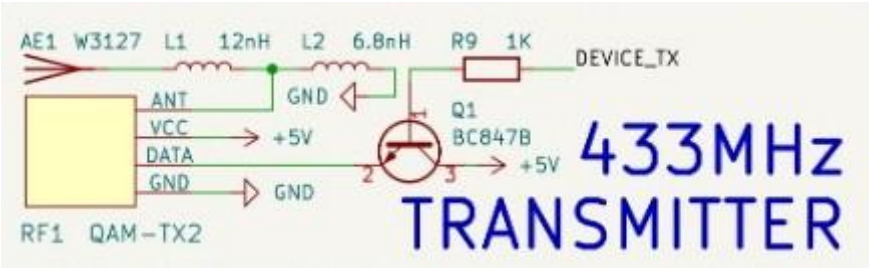


Figure 53: 433MHz transmitter schematic part. *Source: Compiled by the author*.

The component's pin list from its datasheet is seen in the following *Figure 54*.

Pin	Name	Description
1	GND	Ground
2	IN	Data input
3	VCC	Supply Voltage
4	ANT	External Antenna

Figure 54: QAM-TX2-433 pin list. *Source: Quasar UK* [49].

As the receiver counterpart the transmitter circuit is also very simple. The NPN transistor for the input data pin on the transmitter has a 1kΩ resistor (R9) in series with the transistor's base in order to put it in saturation to let the flow of current pass.

Like before, the transmitting antenna has the same matching and tuning circuit as the receiving antenna, a 12nH inductor (L1) in series and a 6.8nH inductor (L2) in parallel.

[42] Vector Network Analyzer.

Real time clock

The following *Figure 55* illustrates the real-time clock part of the electrical schematic. The symbol used already came with the CAD program used.

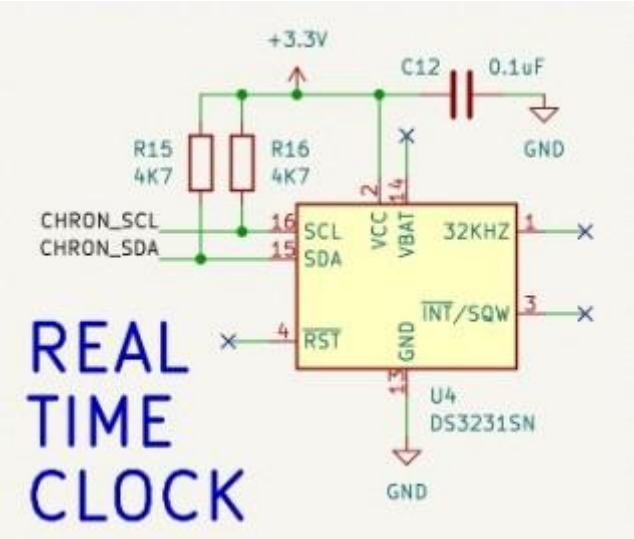


Figure 55: Real time clock schematic part. *Source: Compiled by the author*.

Thanks to the RTC model being specifically selected because of its simplicity and the fact that its battery feature is not used (as already mentioned on the *taken approach* section) the required operating circuit for the chip is almost nonexistent. Following the *DS3231SN typical operating circuit* [51] the chip only needs two 4.7kΩ pull-up resistors (R15, 16) to comply with the I2C interface and a 0.1µF capacitor (C12) to stabilize the power input.

All the other pins of the RTC, listed below in *Figure 56*, are unconnected as the functionalities provided are not used.

PIN	NAME
1	32kHz
2	V_{CC}
3	$\overline{INT}$/SQW
4	$\overline{RST}$
5–12	N.C.
13	GND
14	V_{BAT}
15	SDA
16	SCL

Figure 56: DS3231SN pin list. *Source: Maxim Integrated* [51].

RGB LED

The following *Figure 57* illustrates the RGB LED part of the electrical schematic. The used symbol was designed from the data included in the *RGB LED datasheet* [52].

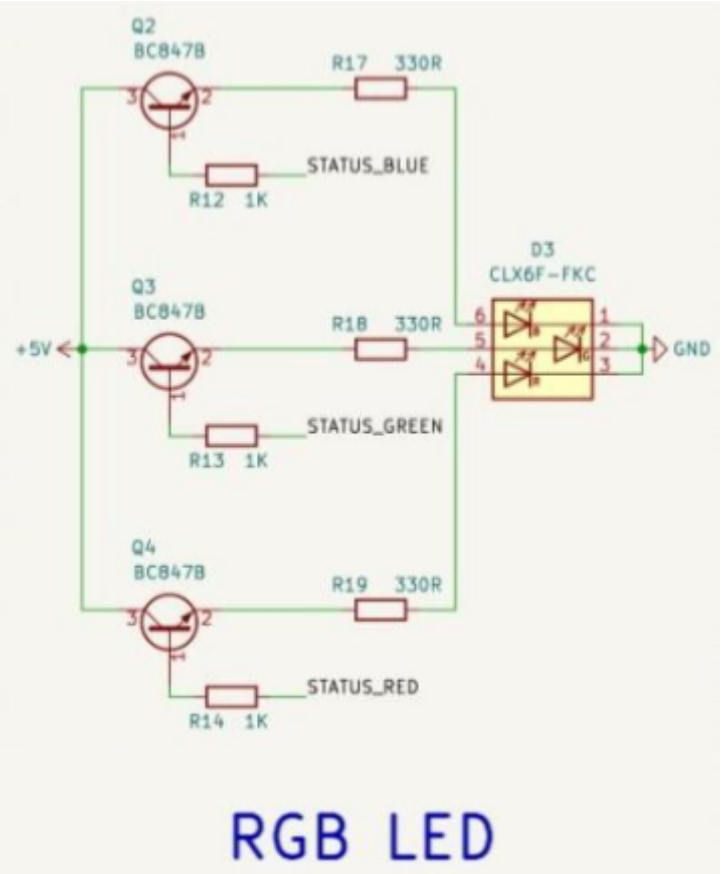


Figure 57: RGB LED schematic part. *Source: Compiled by the author*.

This part has the simplest circuit. An RGB LED is a 3-LED array inside the same cubicle. As the LEDs are of different colors, by additive color theory, any color of the visible spectrum can be generated. The RGB LED's pin list, seen below in *Figure 58*, allows to discern between the red, green and blue color channels.

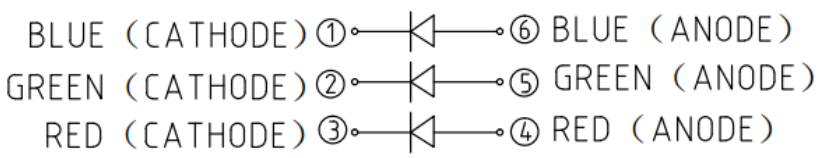


Figure 58: CLX6F-FKC pin list. *Source: Cree* [52].

Each NPN transistor on each LED anode has a 1kΩ resistor (R12~14) in series with the transistor's base in order to put it in saturation to let the flow of current pass. Moreover, in order not to burn the RGB LED, a 330Ω resistor (R17~19) has been placed in series with the anode of each LED.

USB-C connector and protection circuits

The following *Figure 59* illustrates the USB-C connector part and protection circuits of the electrical schematic. All symbols used already came with the CAD program used.

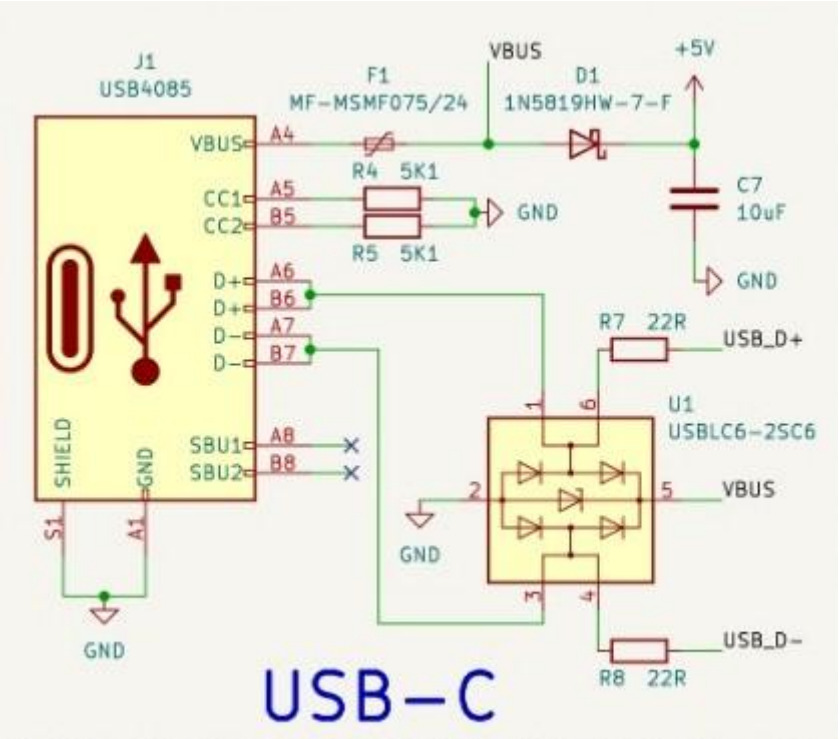


Figure 59: USB-C connector and protection circuits schematic part.
Source: Compiled by the author.

Starting with the USB-C connector all its shielding has been connected with ground in order to reduce ESD spikes.

As the used receptacle follows the USB 2.0 specification, it comes as a 16-pin variant, as seen in the following *Figure 60*, instead of the 24-pin array of common USB-C connectors the USB 3.1 super-speed pins are removed.

Pad No.	1	2	3	4	5	6	7	8	9	10	11	12
Pin No.	A1/B12	A4/B9	A5	B8	B7	A6	B6	A7	B5	A8	B4/A9	A12/B1
Assign	GND	VBUS	CC1	SBU2	D-	D+	D+	D-	CC2	SBU1	VBUS	GND

Figure 60: USB4085 pin list. *Source: GCT* [54].

It must be noticed that pins are duplicated to allow connecting the cable with the receptacle on either side of the port.

Secondary bus pins (SBU1, 2) are directly unconnected as they are intended for an alternate mode which is not required for the given usage.

Configuration channel pins (CC1, 2) serve as cable orientation detection and power delivery negotiation. In this case, the device is intended to receive power from the USB-C port at 5V, therefore the configuration required setup are two 5.1kΩ resistors (R4, 5) in series with such pins and tied to ground.

Both D+ pins and D- pins are connected together for obvious reasons and each one is tied to one of the I/O data input pins of the ESD protection IC. The pin list of the IC used is shown below in *Figure 61*.

Pin name	Pin number	Description
I/O1	1	Data 1 input
GND	2	Ground
I/O2	3	Data 2 input
I/O2	4	Data 2 output
VBUS	5	Power
I/O1	6	Data 1 output

Figure 61: USBLC6-2SC6 pin list. *Source: STMicroelectronics* [59].

The output of such pins is then connected in series with 22Ω termination resistors (R7, 8) that provides (along with correct trace width, length and spacing) impedance matching on the data lines, to ensure that the signals transmitted are properly received and interpreted by the device at the other end of the connection. The value of the resistors is chosen to match the characteristic impedance of the cable (45Ω differential), which helps to minimize reflections and signal loss.

The VBUS pin is the +5V power output of the USB-C receptacle. The first connection is the OC protection polyfuse in order to protect all the other components that follow. Then, it is connected in parallel with the ESD protection IC to reduce voltage spikes. Next is the OV protection schottky rectifying diode in series. Finally, a 10µF capacitor (C7) is used to stabilize the power input which becomes the general +5V power input of the system.

Voltage regulator

The following *Figure 62* illustrates the 3.3V voltage regulator part of the electrical schematic. The used symbol already came with the CAD program used.

As the voltage converter selected is fixed to 3.3V, the circuit is simplified, only requiring a few external capacitors to assure stability to the regulator control loop. The values of such capacitors are taken out of the *LDL1117S33R typical operating circuit* [55] and are two 10µF capacitors (C8, 11), a 1µF capacitor (C9) and a 4.7µF capacitor (C10).

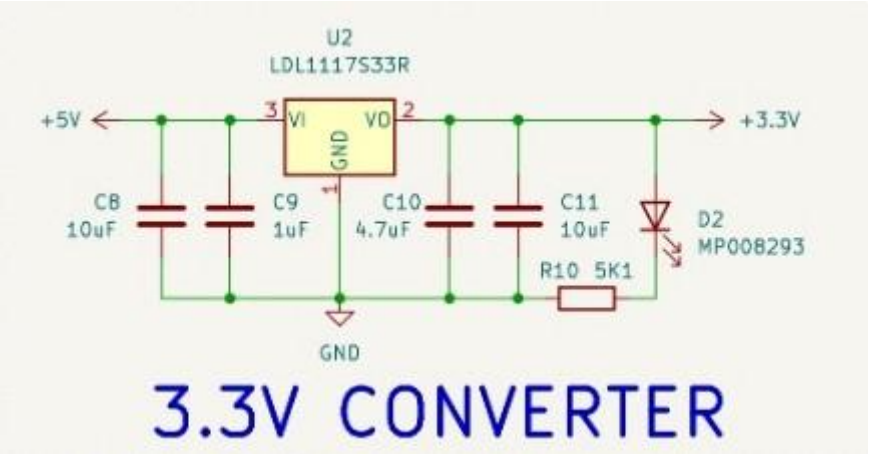


Figure 62: 3.3V voltage regulator schematic part. *Source: Compiled by the author*.

The component's pin list from its datasheet is seen in the following *Figure 63*.

Pin name	Pin number	Description
GND	1	Ground
V_{OUT}	2	Output voltage
V_{IN}	3	Input voltage

Figure 63: LDL1117S33R pin list. *Source: STMicroelectronics* [55].

As mentioned before a red LED is included to show that the device is receiving power. Such LED is connected in parallel to the +3.3V output of the regulator with a 5.1kΩ resistor (R10) in series used to dim its light, as it is not that important and reduces its power consumption.

The power output of the regulator becomes the general +3.3V power input of the system.

Mounting holes

Finally, the following *Figure 64* illustrates the M2 mounting holes part of the electrical schematic. The used symbol already came with the CAD program used.

Figure 64: M2 Mounting holes schematic part. *Source: Compiled by the author*.

10.2. Testing

In order to test the designed electrical schematic before passing onto the PCB design, when the bought hardware components arrived, each different part of the schematic was reproduced in a breadboard such as the 433MHz receiver part shown below in *Figure 65*.

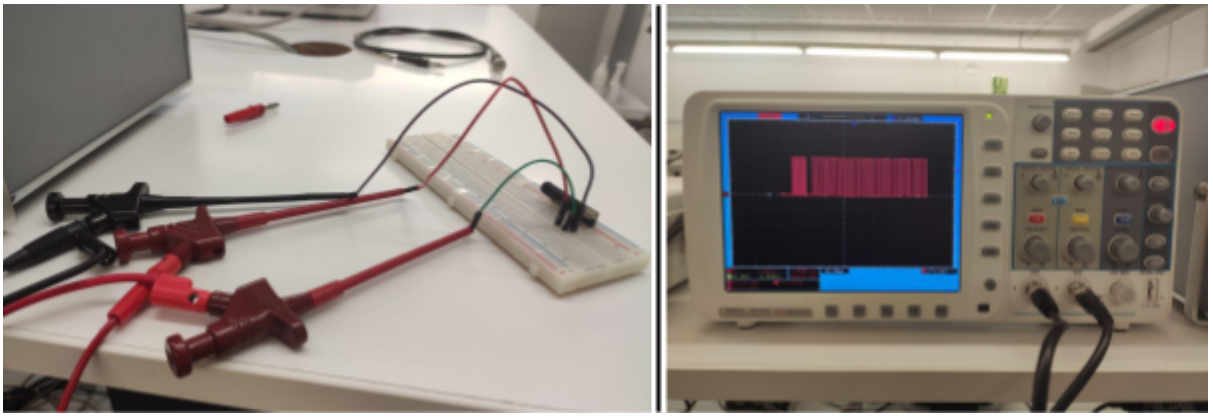


Figure 65: Testing of the 433MHz receiver schematic part. *Source: Compiled by the author*.

In the left side of the figure there is the reproduced receiving circuit and in the ride side there is the oscilloscope output of the data pin of the receiver module. The signal was analyzed to ensure that the module was working correctly, and as seen, a series of changes in the voltage of the pin, marking the width of the sent pulses, is observed when a signal is received by the module.

As mentioned in the *objectives* section of this Thesis, the electrical rule checker (ERC) of the used CAD program was required to pass flawlessly. The following *Figure 66* depicts so.

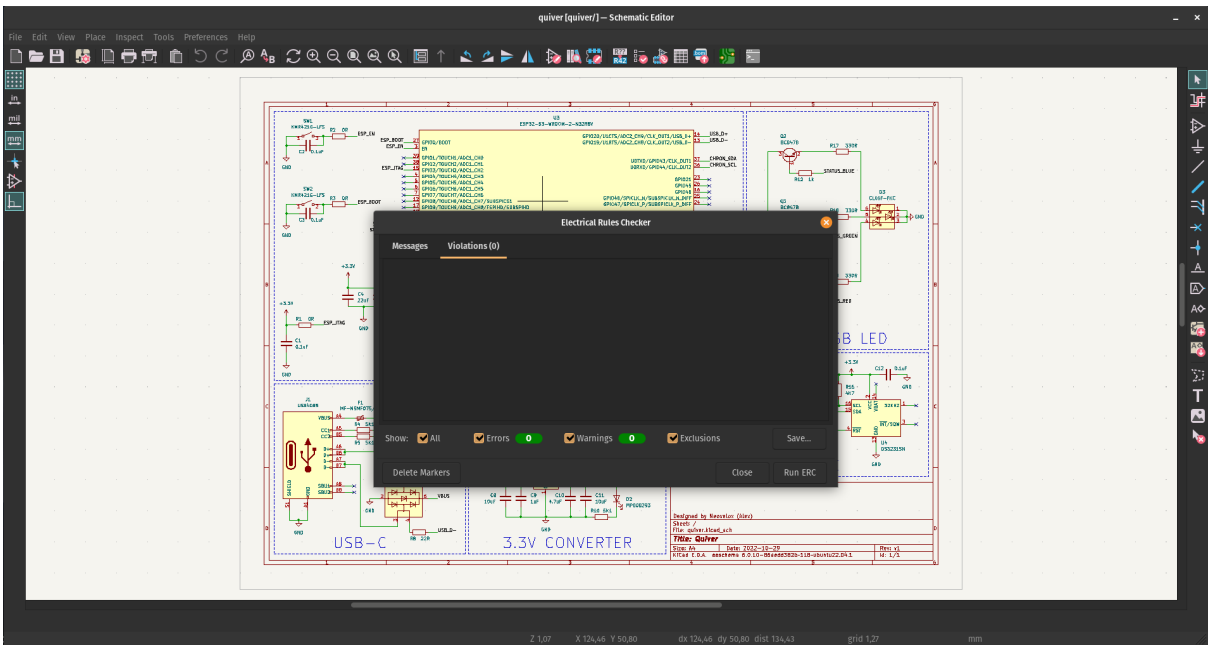


Figure 66: Electrical rule check of the schematic. *Source: Compiled by the author*.

10.3. Problems and solutions

As with any complex design done in limited time without the possibility of iteration, some problems were detected after testing the manufactured PCB. Each problem shown below has a possible solution that was, or was not, taken depending on the severity of the issue.

Emitters are floating when the transistors are cut off.

When the microcontroller output pins driving the different NPN transistors of the schematic, shown below in *Figure 67*, are in a high impedance state because they are not sending anything, the transistors switch to their cut off state. That means, no current flows from the collector to the emitter, thus making the emitter floating.

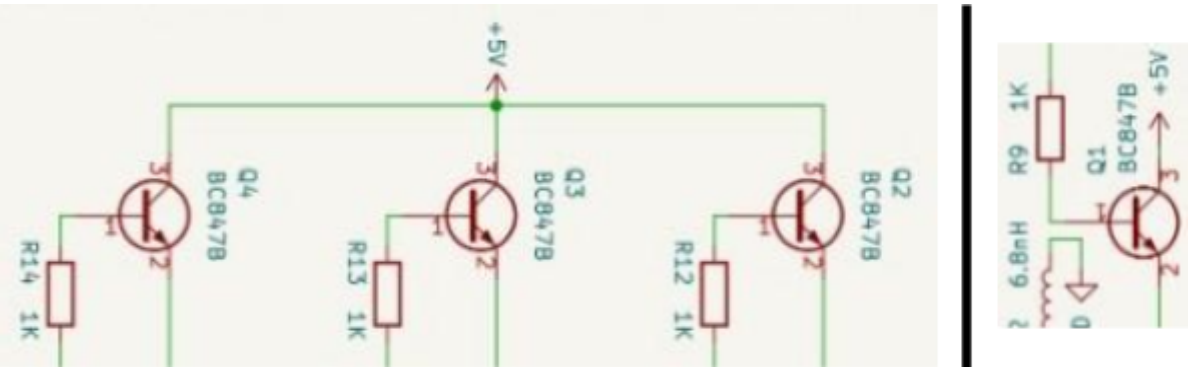


Figure 67: Schematic part of the “emitters are floating when the transistors are cut off” problem. *Source: Compiled by the author*.

The result is that components whose data is driven by such transistors, such as the 433MHz transmitter, may receive wrong data randomly. The severity of this issue is not critical as signals transmitted are very complex and actuators are already equipped with noise cancellation in mind. Moreover, when the transmitter module is sending data, the microcontroller pin is always driving the output pin, ensuring the integrity of the sent command.

A simple yet effective solution would be to put a pull-down resistor on the emitter of each NPN transistor of the schematic.

Microcontroller boots in download mode when powered on.

When the microcontroller is powered on for the first time, that is not resetting the chip with the push button, it boots directly to the download mode getting stuck waiting for firmware to be flashed. Resetting the chip once with the included push button makes the microcontroller to now boot directly to the firmware instead.

That happens because of the capacitor (C3) shown below in *Figure 68*. Remember that the ESP_BOOT (GPIO0) strapping pin of the chip selects the booting mode, as shown in the previous *Table 10* of this Thesis.

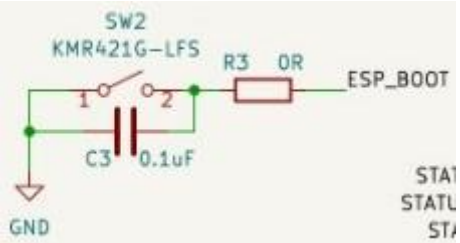


Figure 68: Schematic part of the "microcontroller boots in download mode when powered on" problem. *Source: Compiled by the author*.

Although, the default boot mode of the chip is firmware by having a weak pull-up resistor, when the whole board is first powered on the capacitor is not charged and therefore starts to charge. In all the time that it takes to charge the capacitor to at least a logic 1, the pin is in a logic 0, thus configuring the strapping pin to boot to the download mode instead.

After a chip reset, as the board is already powered on the capacitor is already charged, the pin is always in a logic 1 thanks to the internal weak pull-up resistor, thus configuring the strapping pin to boot to the firmware mode this time.

The severity of this issue is not critical in a development environment, but when the board is installed inside the casing the push buttons cannot be accessed, making it impossible to reset the chip without powering off the system and therefore discharging the problematic capacitor.

The simple yet effective solution taken was to just desolder the offender capacitor. As its usage is just for debouncing the button, there are no consequences when removing it.

Transmission antenna does not send anything.

This is a problem caused by two things, the other cause is explained later on in the *PCB* section of this Thesis.

Although proving that the microcontroller is actually sending data to the transmitter module by attaching it to an oscilloscope and analyzing it, the transmitter or the antenna does not seem to send anything, as tested with the RTL-SDR scanner.

One cause to this problem was the inductor (L2) shown below in *Figure 69*.

Figure 69: Schematic part of the "transmission antenna does not send anything" problem.
Source: Compiled by the author.

Because of lack of antenna design knowledge, the matching and tuning circuit from the receiving antenna was just replicated in the transmitting antenna without thinking of its consequences.

For the reception antenna this inductor connected to ground helps filtering frequencies lower than the 433MHz band, thus providing some sort of noise cancellation.

However, inductors have very low resistance (in the milli magnitude), and therefore low impedance. Because of that, the current from the antenna pin of the transmitter module tends to flow to ground through the inductor as is the path with least impedance, and hence, no current hits the antenna.

The simple yet effective solution taken was to just desolder the offender inductor. As its usage is just for the antenna's matching and tuning circuit, there are no consequences when removing it.

10.4. Results

The final complete electrical schematic design is included below in *Figure 70*.

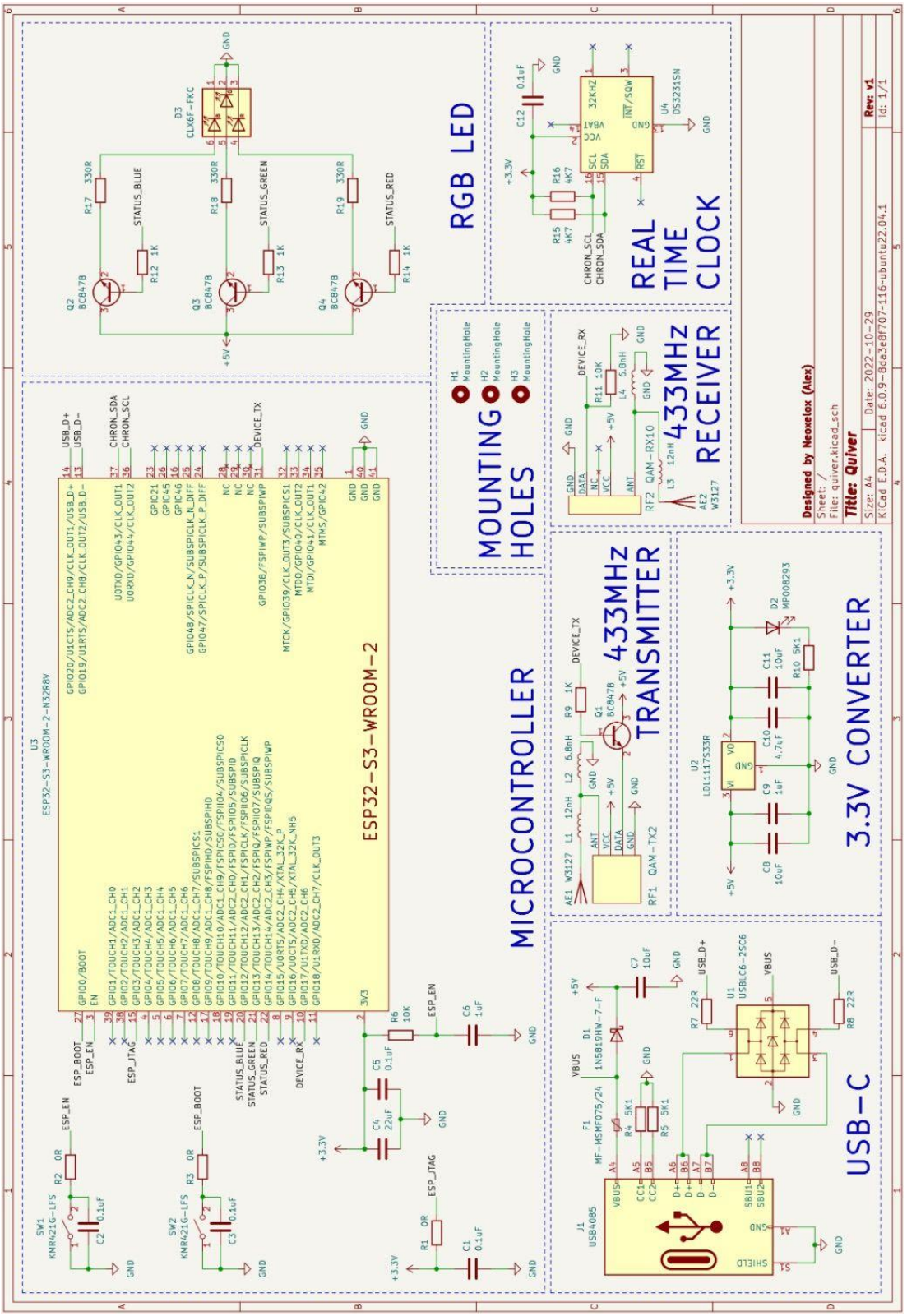


Figure 70: Complete electrical schematic. *Source: Compiled by the author*.

11. PCB

11.1. Design

Microcontroller

The following *Figure 71* illustrates the microcontroller part of the PCB. The used footprint was given by the vendor. As the two push buttons are a little far away from the microcontroller they have been included separately on the right.

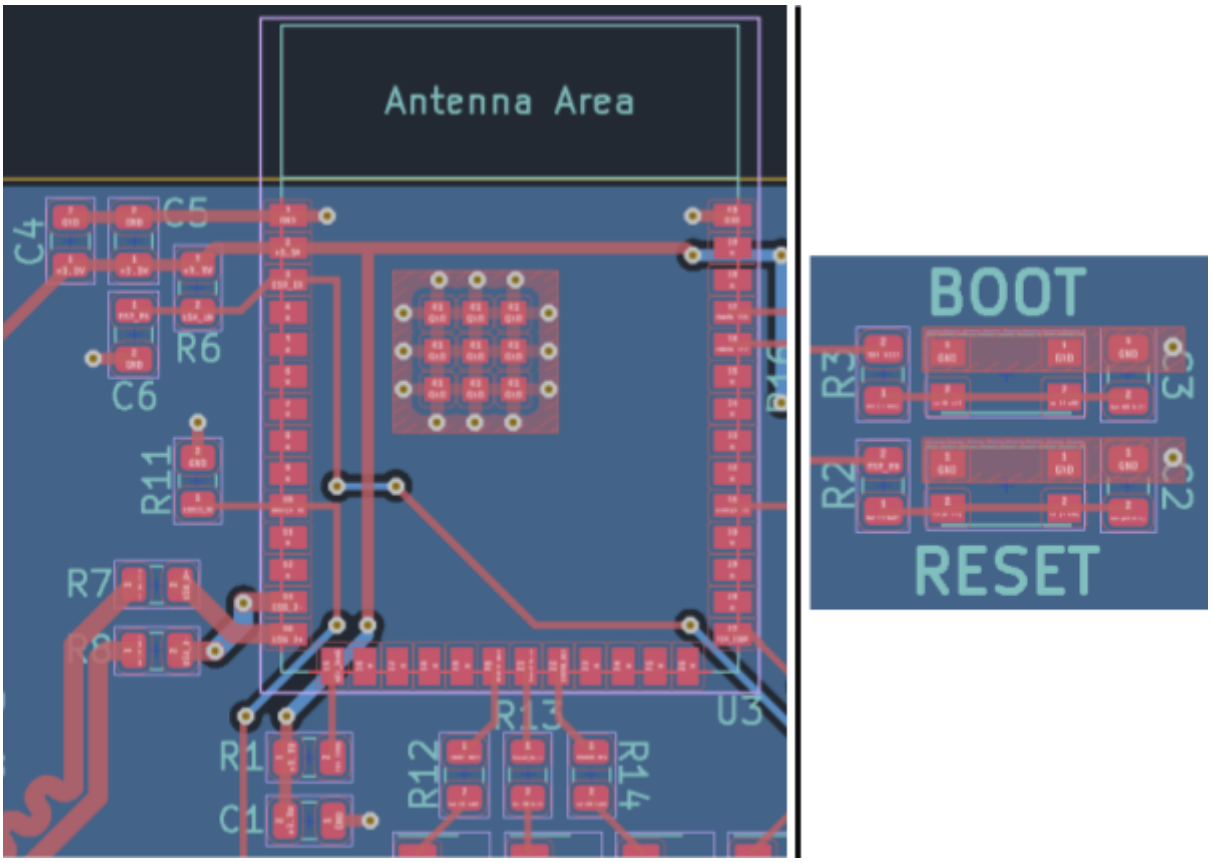


Figure 71: Microcontroller PCB part. *Source: Compiled by the author*.

All the ground pins underneath the chip (EPAD) have been connected together with a little copper pour and a lot of vias to connect them to the bottom ground plane as short as possible, avoiding possible ground loops and therefore decreasing potential noise interference.

The power on sequence RC delay circuit (R6, C6) has been placed very close to the +3.3V and the reset pin (EN) of the chip, to avoid increasing the impedance, as well as the power stabilizing capacitors (C4, 5). The capacitor (C1) stabilizing the JTAG strapping pin (GPIO3) has been placed close to it for the same reason.

The following *Figure 72* illustrates the pin layout of the microcontroller. As said, in order to reduce routing complexity, the other components have been placed to the side corresponding with their used pins on the chip.

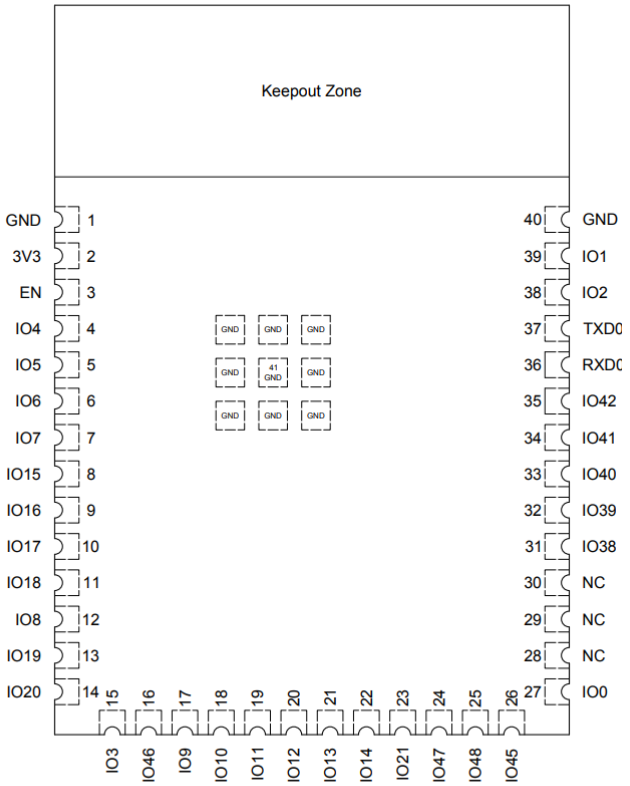


Figure 72: ESP32-S3-WROOM-2-N32R8V pin layout. *Source: Espressif* [64].

Following the *ESP32-S3 series hardware design guidelines* [65], for a 2-layer PCB a complete ground plane should be provided on the bottom without any other components and keeping the traces to a minimum. The top layer can be used for signal traces and components but another ground plane is not advised as it could generate possible crosstalks between very close traces.

Moreover, the chip's antenna should be placed outside the PCB in order to reduce interference from the board, as seen in the keepout zone of *Figure 73* below.

Moving on to the push buttons, their debouncing capacitors (C2, 3) have been placed as close as possible to avoid increasing the impedance. Moreover, texts describing their use have been included as silkscreen, which will be printed on top of the PCB's solder mask.

Finally, the USB data line termination resistors (R7, 8) are placed as close as possible to the chip USB pins (GPIO19, 20) to minimize the effects of signal reflections. These reflections can cause an impedance mismatch resulting in a distortion of the data signal.

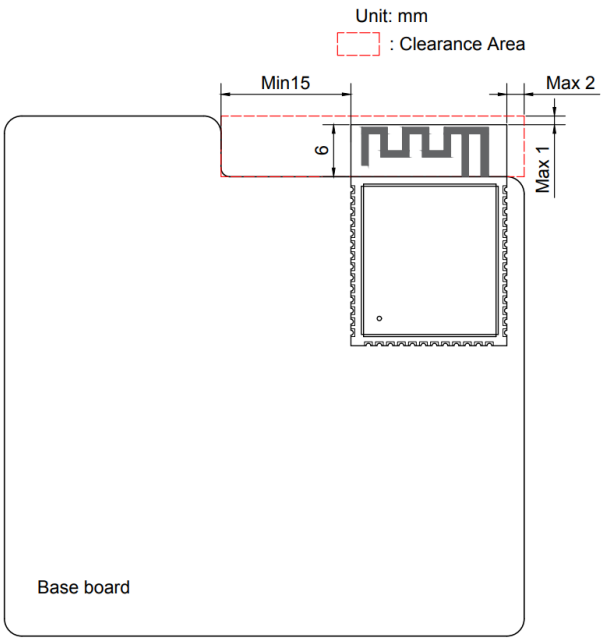


Figure 73: Keepout zone for ESP32-S3 series module's antenna. *Source: Espressif* [65].

Receiver

The following *Figure 74* illustrates the 433MHz receiver part of the PCB. The used footprint was designed from the data included in the *QAM-RX10-433 datasheet* [48].

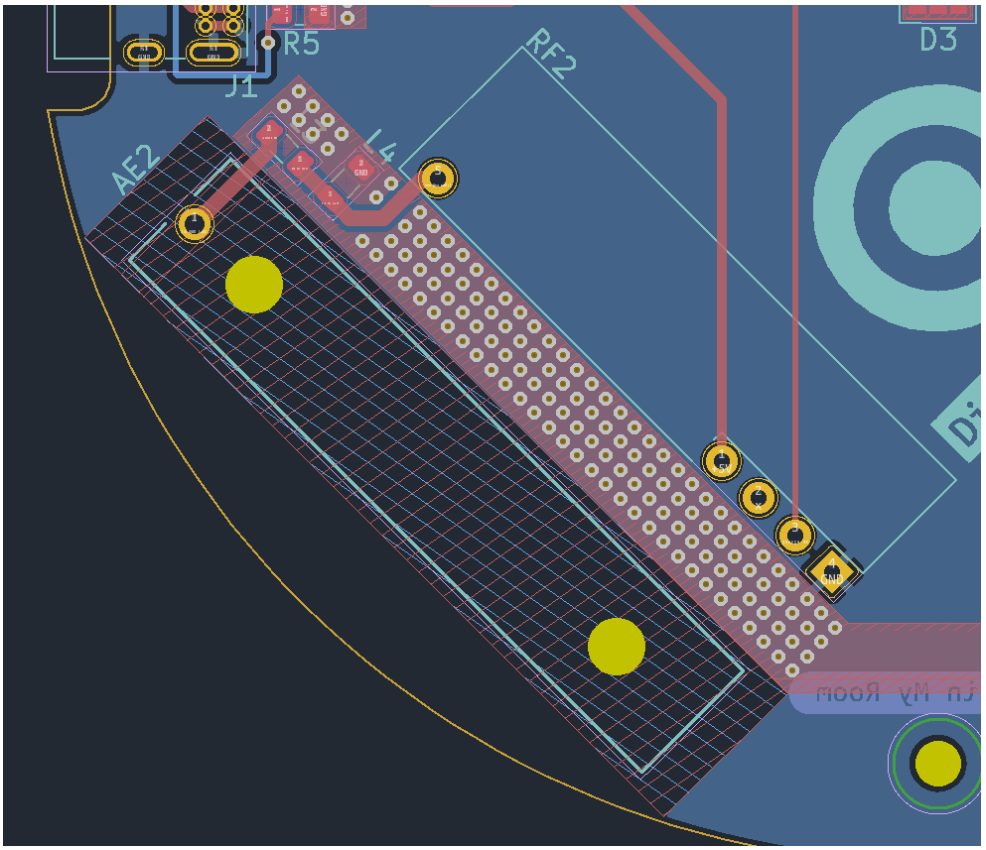


Figure 74: 433MHz receiver PCB part. *Source: Compiled by the author*.

As the receiver comes as a module it will be soldered to the PCB as a through hole component. Although the pin layout from its datasheet, seen in *Figure 75*, does not illustrate the antenna pin, in reality is on the opposite side of all other pins.

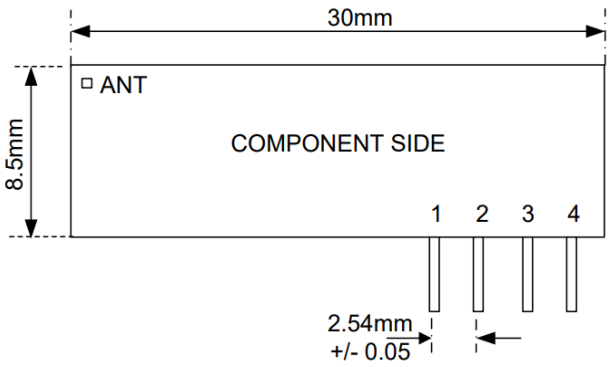


Figure 75: QAM-RX10-433 pin layout. *Source: Quasar UK* [48].

The receiver antenna is placed next to the module to reduce increasing the impedance and on the opposite sides of the microcontroller's and transmitter's antennas to avoid antenna coupling. Moreover, following the *W3127 antenna hardware design guidelines* [50], the antenna must have a ground clearance area from all PCB layers and should be surrounded by a grounded via fence, as shown in the following *Figure 76*.

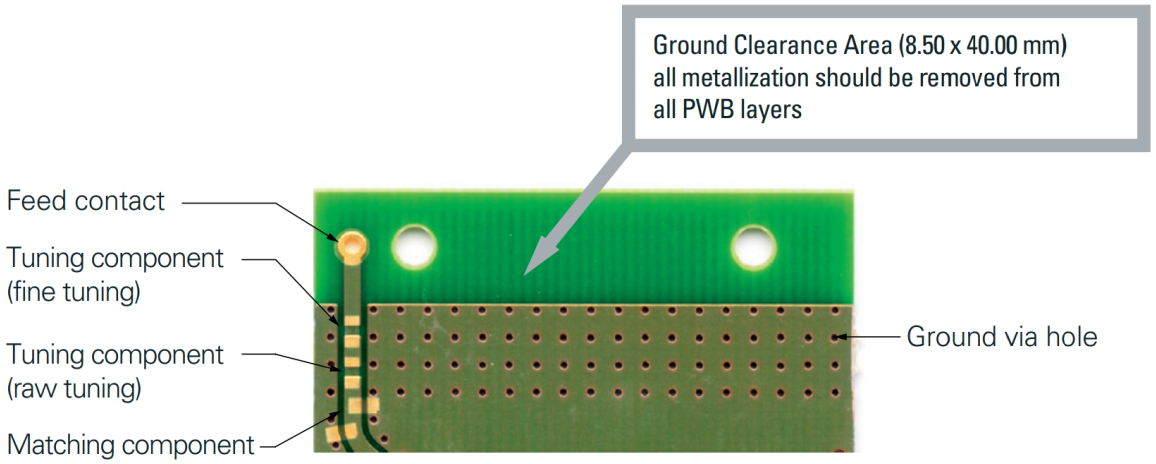
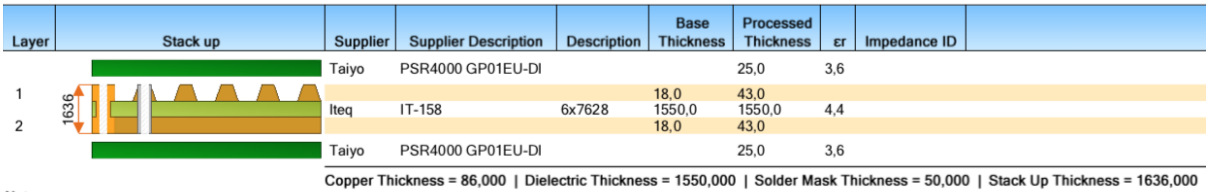


Figure 76: W3127 PCB layout. *Source: Compiled by the author*.

Matching and tuning components (L3, 4) should be placed in the same order and orientation seen in the previous PCB layout guide.

Another critical thing to take into account for the 433MHz antennas is to assert impedance matching on the antenna traces to ensure signal integrity. In order to do so the length, width and spacing of the routed traces have to be correctly configured.

In order to calculate such variables, several parameters such as the copper thickness or the dielectric constant have to be gathered from the vendor that will manufacture the PCB. The following *Figure 77* depicts the information of all the layers that form a 2-layer PCB from Aisler.



Layer	Stack up	Supplier	Supplier Description	Description	Base Thickness	Processed Thickness	εr	Impedance ID
		Taiyo	PSR4000 GP01EU-DI			25,0	3,6	
1					18,0	43,0		
	1636	Iteq	IT-158	6x7628	1550,0	1550,0	4,4	
2					18,0	43,0		
		Taiyo	PSR4000 GP01EU-DI			25,0	3,6	

Copper Thickness = 86,000 | Dielectric Thickness = 1550,000 | Solder Mask Thickness = 50,000 | Stack Up Thickness = 1636,000

Figure 77: Aisler's 2-layer PCB parameters. *Source: Aisler* [15].

The formulas used to compute the correct length, width and spacing for routing the traces are very complex and depend on several variables. As this resembles more to an electrical engineering aspect and I lack the knowledge, formulas are not described as go out of the Thesis scope. Luckily, the CAD program used, KiCad, has a built-in calculator for such cases as shown below in *Figure 78*. Knowing that the routed trace is only one (not differential) and that there is a ground plane on the bottom layer, we can select our transmission line type to "coplanar waveguide with ground plane" and perform the analysis.

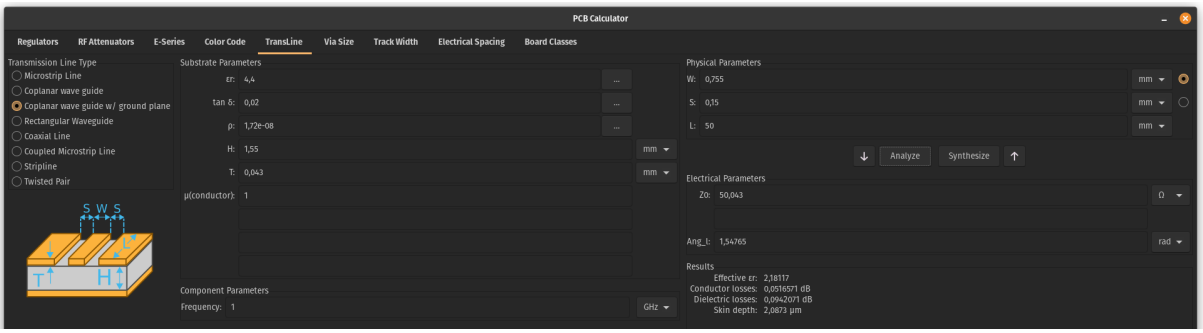


Figure 78: KiCad's impedance matching calculator for the 433MHz antenna lines. *Source: KiCad* [36].

The resulting parameters for the required 50Ω impedance matching are a 0.75mm trace width, a 0.15mm maximum spacing and 50mm of maximum length.

Transmitter

The following *Figure 79* illustrates the 433MHz transmitter part of the PCB. The used footprint was designed from the data included in the *QAM-TX2-433 datasheet* [49].

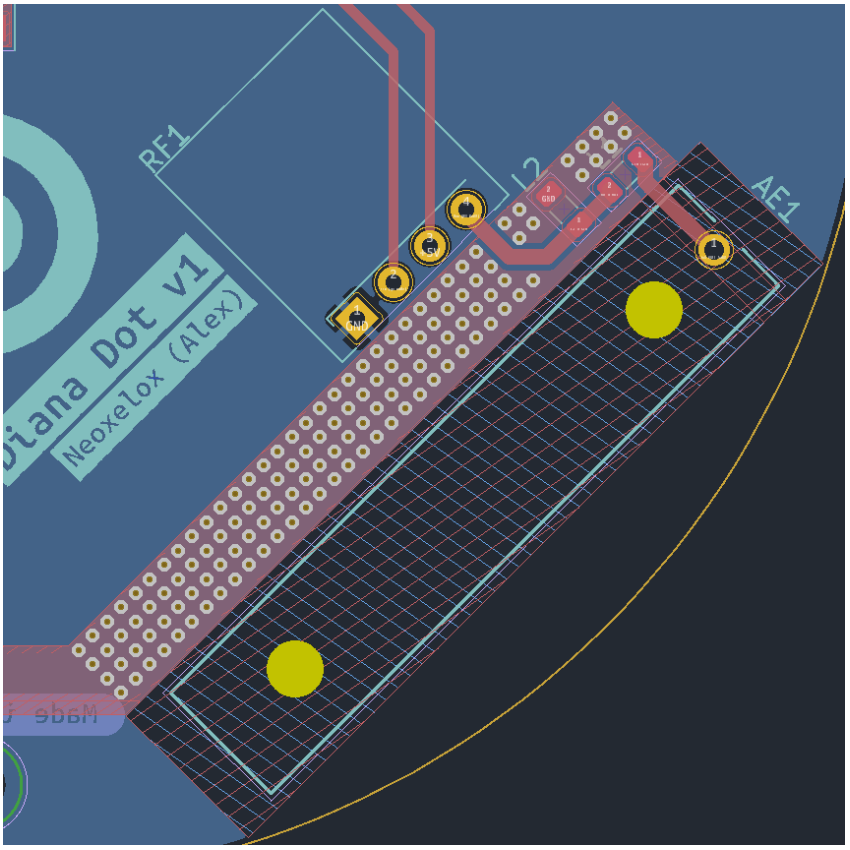


Figure 79: 433MHz transmitter PCB part. *Source: Compiled by the author*.

The transmitter also comes as a module so it will be soldered to the PCB as a through hole component. The transmitting antenna is also placed next to the module to reduce increasing the impedance and on the opposite sides of the microcontroller's and receiver's antennas to avoid antenna coupling. The PCB layout requirements for this antenna are the same as the antenna of the receiving module: ground clearance area from all PCB layers, grounded via fencing surrounding it and 50Ω impedance matching on the antenna pin.

The component's pin layout from its datasheet is seen in the following *Figure 80*.

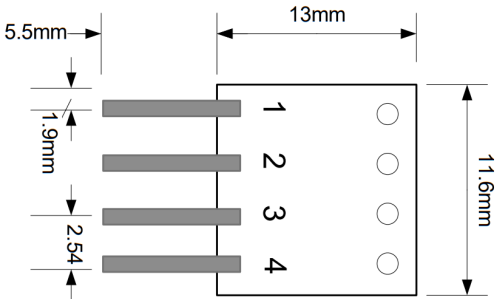


Figure 80: QAM-TX2-433 pin layout. *Source: Quasar UK* [49].

Real time clock

The following *Figure 81* illustrates the real-time clock part of the PCB. The footprint used already came with the CAD program used.

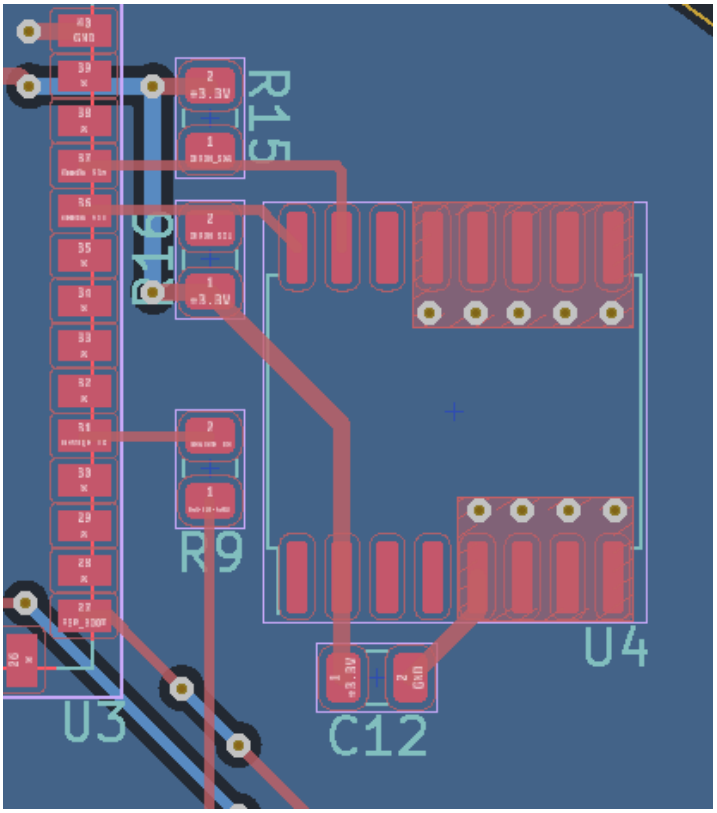


Figure 81: Real time clock PCB part. *Source: Compiled by the author*.

The pull-up resistors (R15, 16) are placed close to their I2C respective lines to avoid increasing the impedance and the stabilizing capacitor (C12) is next to the +3.3V power input pin of the IC. The chip's not-connected pins (NC) are all tied together with a little copper pour and a lot of vias to connect them to the bottom ground plane as short as possible, avoiding possible ground loops and therefore decreasing potential noise interference.

The component's pin layout from its datasheet is seen in the following *Figure 82*.

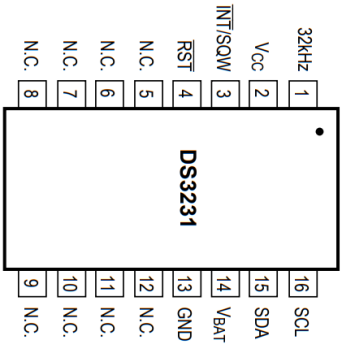


Figure 82: DS3231SN pin layout. *Source: Maxim Integrated* [51].

RGB LED

The following *Figure 83* illustrates the RGB LED part of the PCB. The used footprint was designed from the data included in the *RGB LED datasheet* [52].

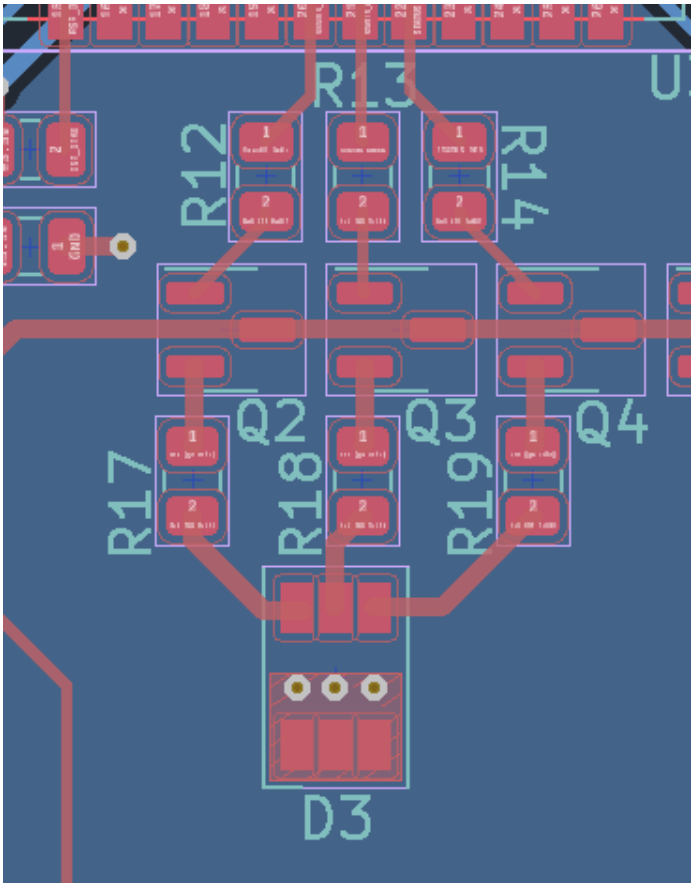


Figure 83: RGB LED PCB part. *Source: Compiled by the author*.

All components forming the RGB LED circuit are very close to each other not only because of impedance reasons, but also because the LED must be at the center of the PCB, so it aligns correctly with the casing center hole, and in order to not waste unnecessary space the microcontroller has not been placed further away. The LEDs cathodes are all tied together with a little copper pour and a lot of vias to avoid ground loops.

The component's pin layout from its datasheet is seen in the following *Figure 84*.

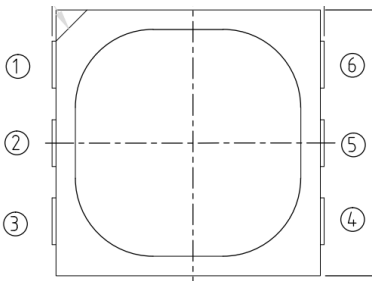


Figure 84: CLX6F-FKC pin layout. *Source: Cree* [52].

USB-C connector and protection circuits

The following *Figure 85* illustrates the USB-C connector part and protection circuits of the PCB. All footprints used already came with the CAD program used.

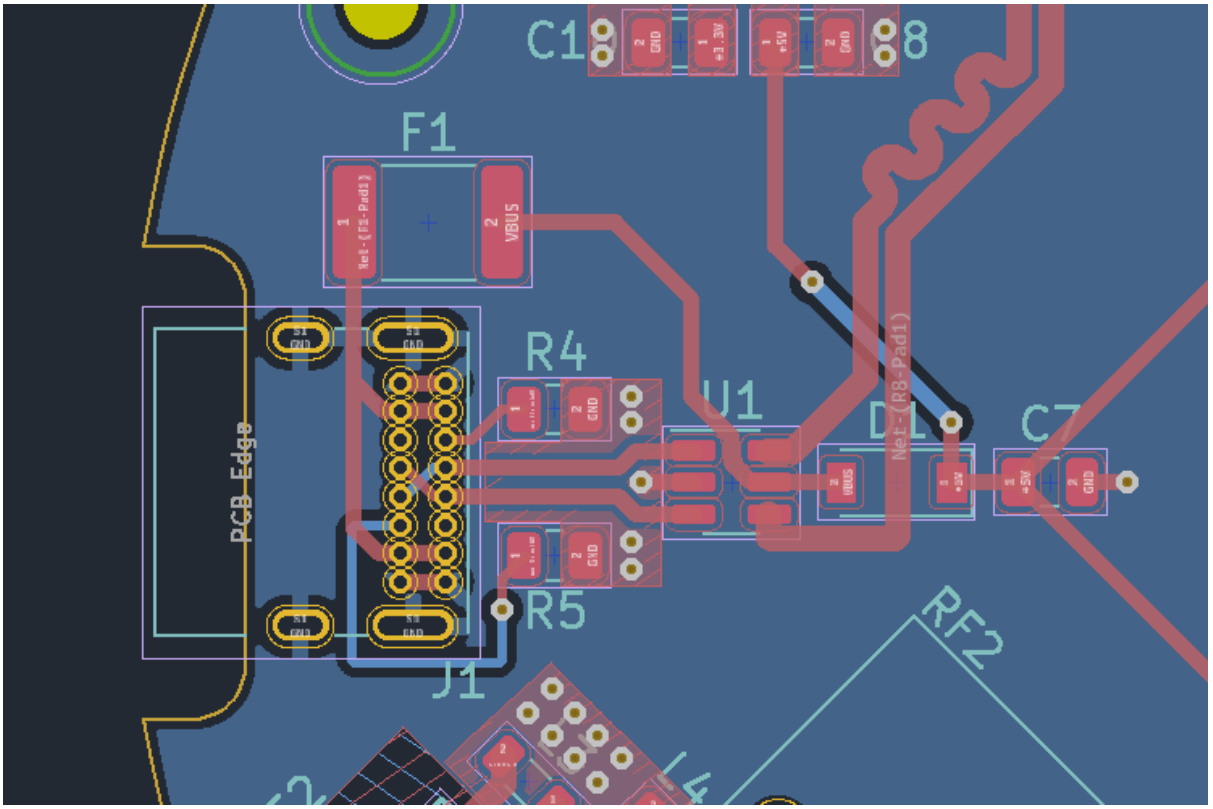


Figure 85: USB-C connector and protection circuits PCB part.
Source: Compiled by the author.

Starting with the PCB outline, as decided before, in the *taken approach* section, no component must lay out of the board in order to achieve a low side profile. That's why there is an indentation for the USB-C receptacle. Most of the pins of the connector have been tied together as they are duplicated in order to allow to connect the cable to the receptacle on any side, as seen below in *Figure 86*. The configuration resistors on the CC lines have been placed together very close to their respective pins to avoid increasing impedance and mismatching the required configuration.

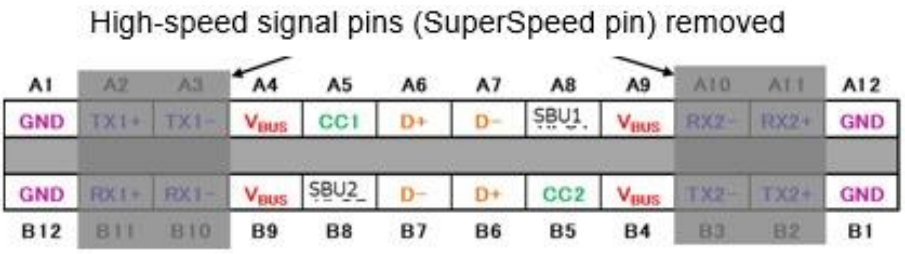


Figure 86: USB4085 pin layout. *Source: GCT* [54].

The OC protection polyfuse is placed separately from the other components since it can be subject to high temperatures. The ESD protection IC is placed next following the pin layout from its datasheet as seen in the following *Figure 87*. The next component is the OV protection schottky rectifying diode, and finally the stabilizing capacitor (C7).

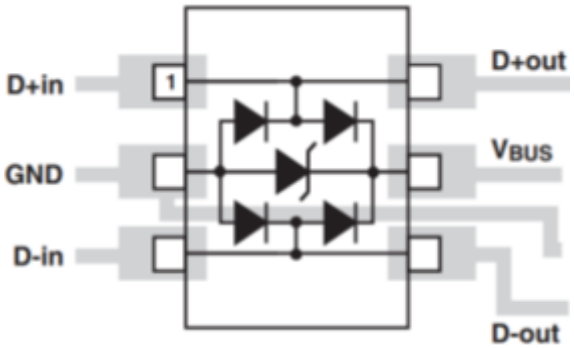


Figure 87: USBLC6-2SC6 pin layout. *Source: STMicroelectronics* [59].

The USB-C differential data line (D+, D-) is required to be routed together to both: ensure matching the differential impedance required, 45Ω; and ensure matching the length on both traces so that the differential signal can arrive at the same time.

As the data traces are separated by the VBUS line on the ESD protection IC the routing became a little bit complex. In order to ensure matching both lengths three meanders were routed on the D+ line to increase its length and match the D- line.

The routing parameters for the differential impedance matching were calculated with the KiCad calculator, as seen below in *Figure 88*, the same way as in the *receiver* section. The type of the transmission line in this case is "coupled microstrip line".

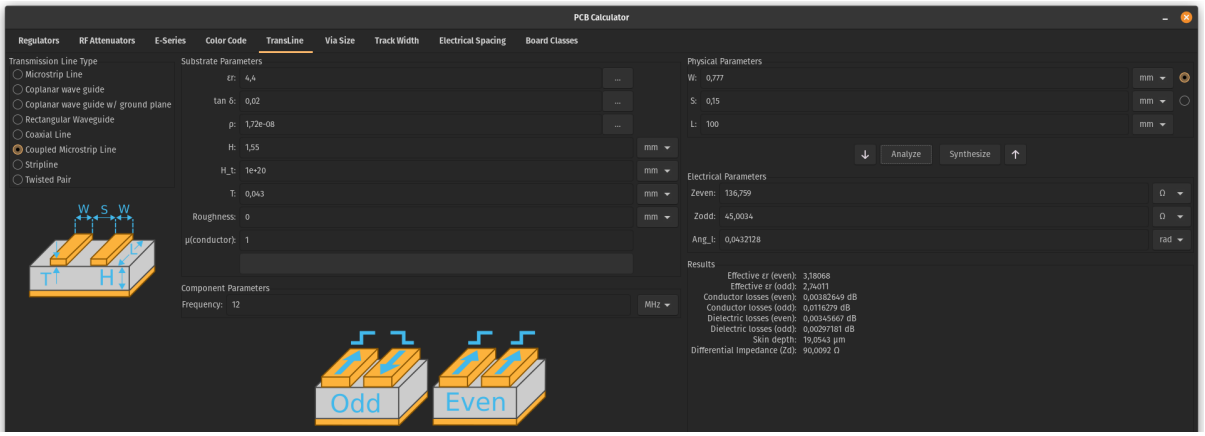


Figure 88: KiCad's impedance matching calculator for the USB-C 2.0 data lines. *Source: KiCad* [36].

The resulting parameters for the required 45Ω differential impedance matching are a 0.77mm trace width, a 0.15mm spacing and 100mm of maximum length.

Voltage regulator

The following *Figure 89* illustrates the 3.3V voltage regulator part of the PCB. The used footprints already came with the CAD program used.

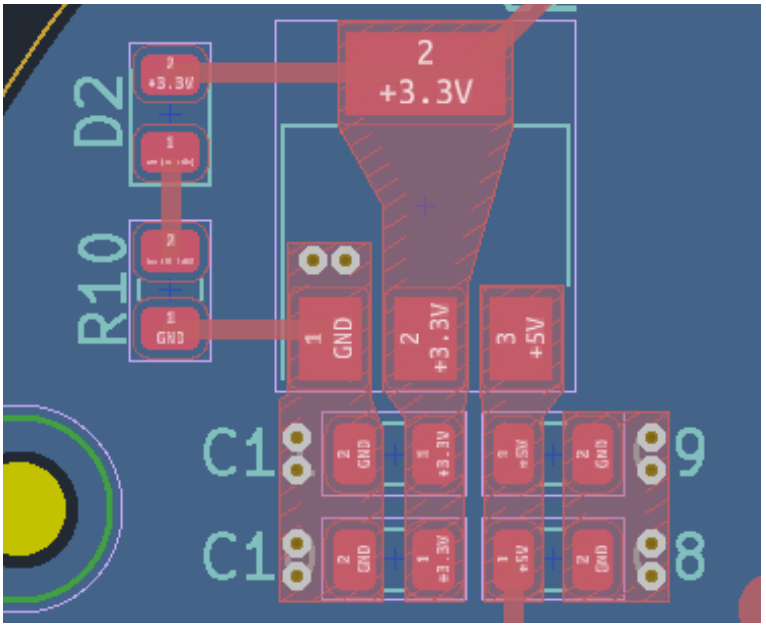


Figure 89: 3.3V voltage regulator PCB part. *Source: Compiled by the author*.

The red LED is placed close to the PCB edge to avoid interfering with the light from the central RGB LED. Stabilizing capacitors (C8~11) are placed close to their respective power input and output to calibrate the regulator control loop. The regulator's pin layout from its datasheet is seen in the following *Figure 90*.

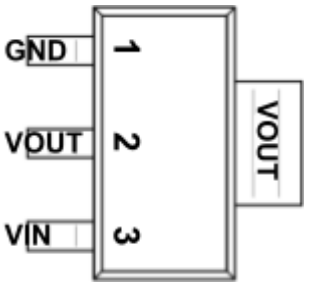


Figure 90: LDL1117S33R pin layout. *Source: STMicroelectronics* [55].

Mounting holes

Finally, the following *Figure 91* illustrates one of the three M2 mounting holes part of the PCB. The used footprint already came with the CAD program used. In order to see the disposition of all three mounting holes please refer to the complete PCB design on *Figure 105*.

Figure 91: M2 Mounting holes PCB part. *Source: Compiled by the author*.

3D preview

As stated in the *requirements* section of this Thesis, a 3D preview of the PCB without any missing hardware component should be included. It allows to preview how the final manufactured PCB, holes and indentations will look like, helping to debug overlapping, positioning and measuring problems and issues with the board overall outline shape. Moreover, having a 3D model of the PCB is a great resource in order to design flawlessly the casing later on.

The following *Figure 92* depicts such 3D render. A large version of the 3D render in an angle view has been included later on the *PCB results* section.

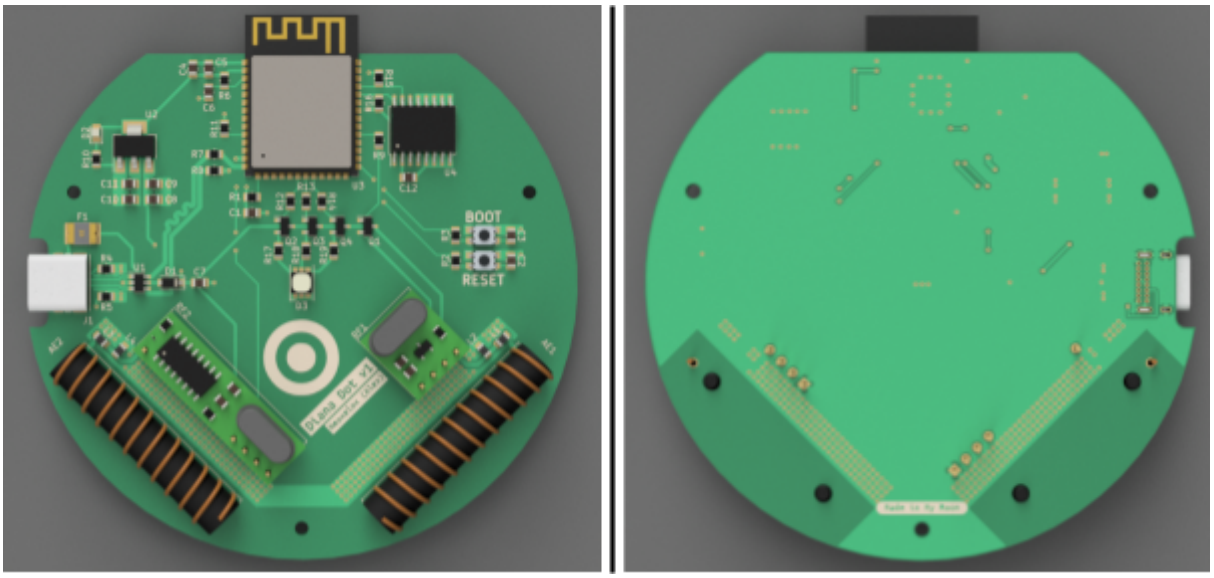


Figure 92: PCB 3D render top and bottom view. *Source: Compiled by the author*.

The CAD program used, KiCad, already came with 3D models for resistors, capacitors, inductors, LEDs and common IC packages. PCB outline and traces were also automatically created by KiCad.

The microcontroller, USB-C receptacle, polyfuse, RGB LED and the push buttons models were given by their respective vendors.

The antennas and the 433MHz modules were created from scratch following the features and somewhat sub-components of their respectives datasheets.

All hardware components were put together in Fusion360 and its render tool was used to produce the image.

11.2. Manufacturing

Board

The first step taken to manufacture the PCB was to reach Aisler to print the designed board. It can be done through their website and a compressed folder containing all gerber files for all layers definitions must be uploaded as it can be seen below in *Figure 93*.

After the upload, the application presents us with the PCB overview and automatic tests passed to ensure the board is actually manufacturable with Aisler's standards. Depending on the complexity of the PCB the price increases, in this case the design rule rating given to the board was "complex".

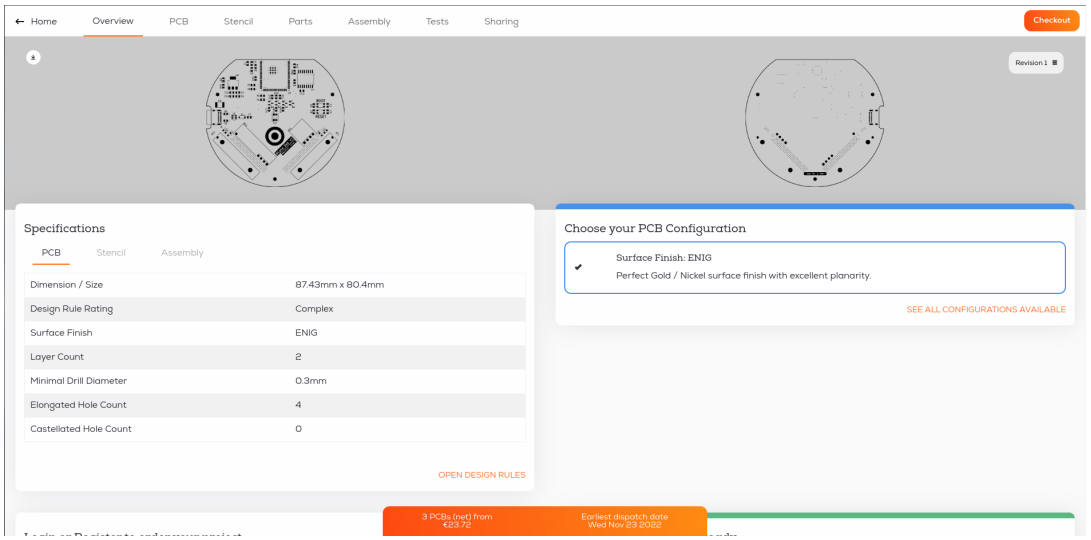


Figure 93: Aisler PCB order overview. *Source: Compiled by the author*.

Aisler also includes a preview on how the PCB will look like, to ensure complex designed shapes and holes are manufactured correctly. The following *Figure 94* depicts so.

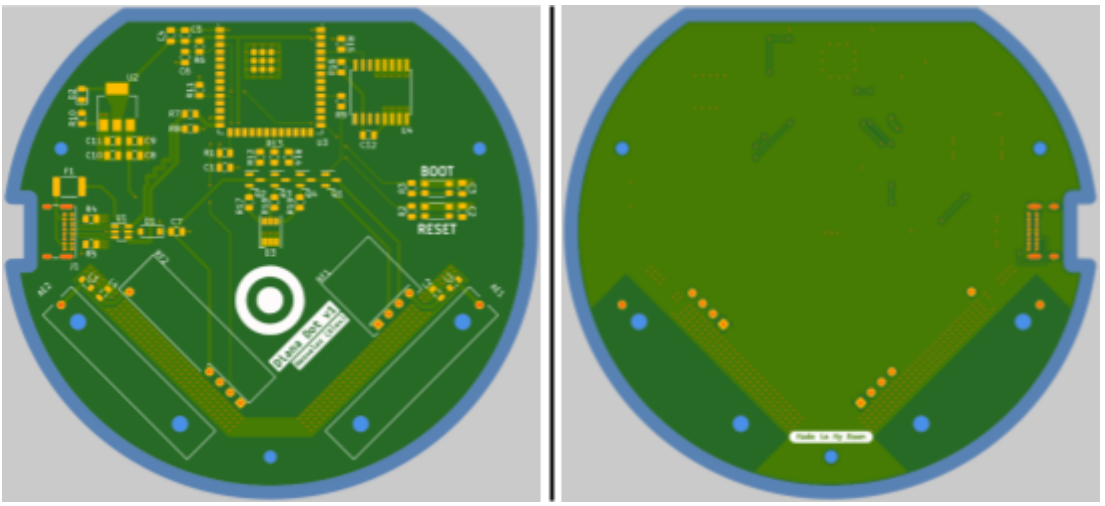


Figure 94: Aisler PCB preview top and bottom views. *Source: Compiled by the author*.

As the selected vendor is from the European Union, order to arrival took less than 2 weeks. The different manufactured PCB samples came panelized as shown below in *Figure 95*. Before continuing to the soldering step, the PCB was taken out from the panel with some pliers.

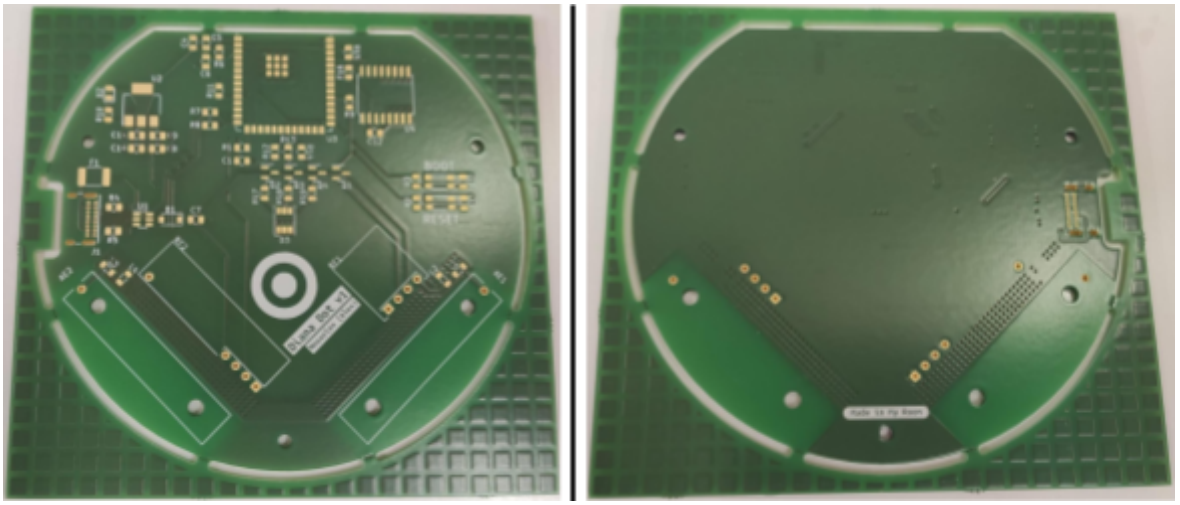


Figure 95: Aisler PCB manufactured top and bottom views. *Source: Compiled by the author*.

SMD components

The board features both SMD and THT components, and as decided the SMD parts will be soldered using an electronics oven. The first step taken was to apply soldering paste to the pads of the SMD components, which only reside on the top layer of the PCB. Next, all components must be placed carefully on top of their pads one by one while doing a little bit of pressure. The following *Figure 96* depicts such process.

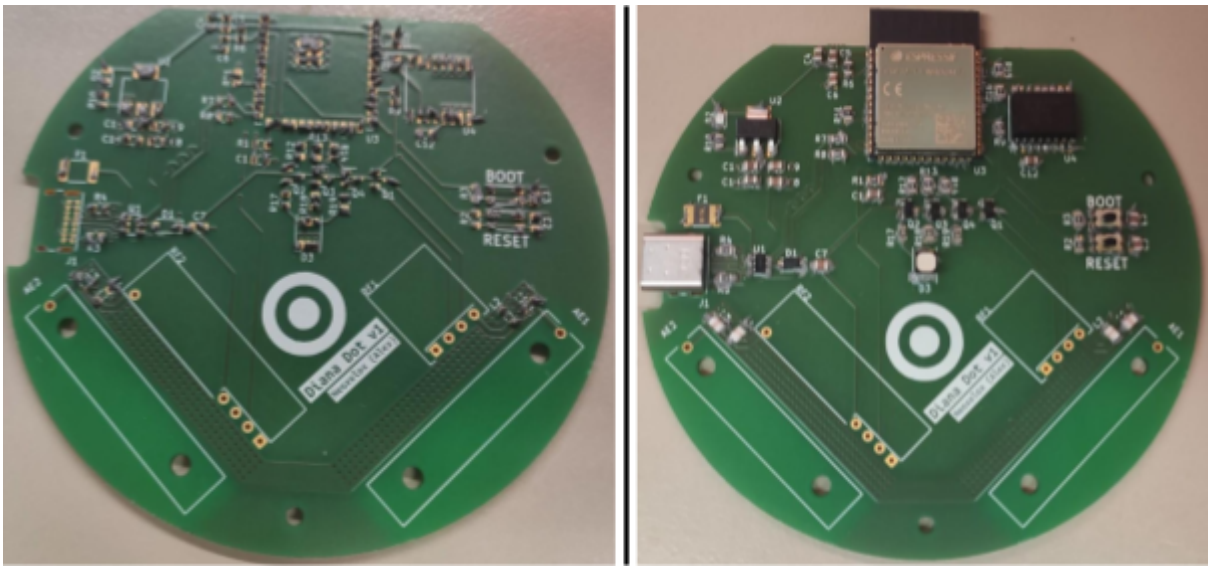


Figure 96: SMD components placed with solder paste. *Source: Compiled by the author*.

The next step was to use the electronic oven shown below in *Figure 97* to melt that solder paste, soldering all the SMD components to the board.

Figure 97: Electronics oven used. *Source: Compiled by the author*.

The reflow profile used to solder the components, that is the soldering temperature and resting periods, was the profile from the microcontroller, shown in the following *Figure 98*, as it is the most critical component in the PCB. If the profile is not followed it could potentially damage the chip because of the humidity inside of it.

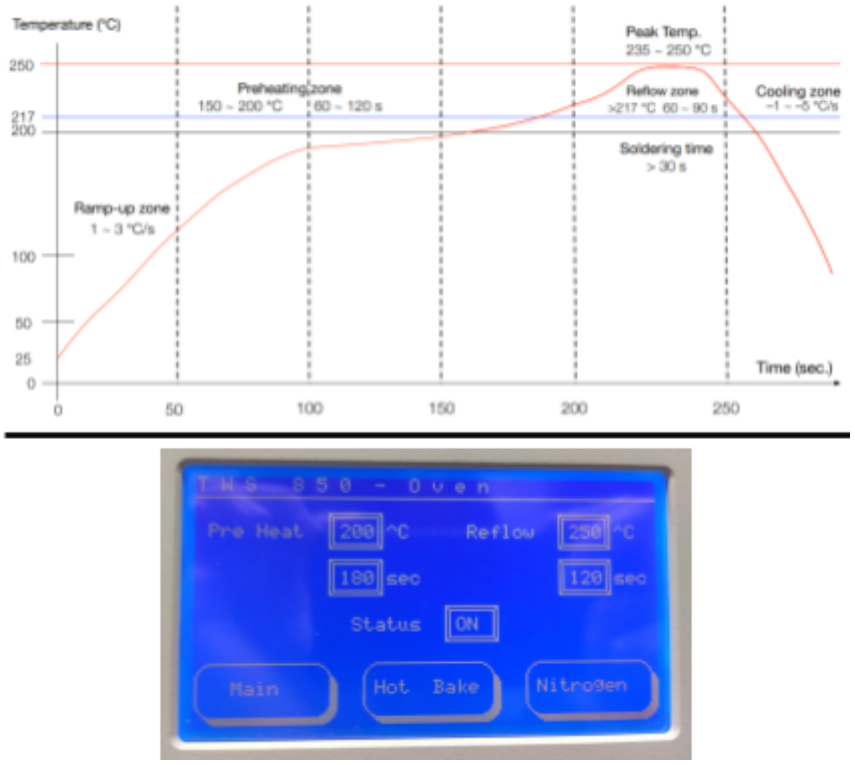


Figure 98: ESP32-S3-WROOM-2 reflow profile on the top, electronics oven configuration on the bottom. *Source: Espressif* [47] *and compiled by the author*.

Below in *Figure 99* there is the result of the PCB with the SMD components soldered. The board out of the oven was not perfect, several components were misaligned and had shorts

within their connections. With the aid of a multimeter, flux and a soldering iron, the issues were solved.

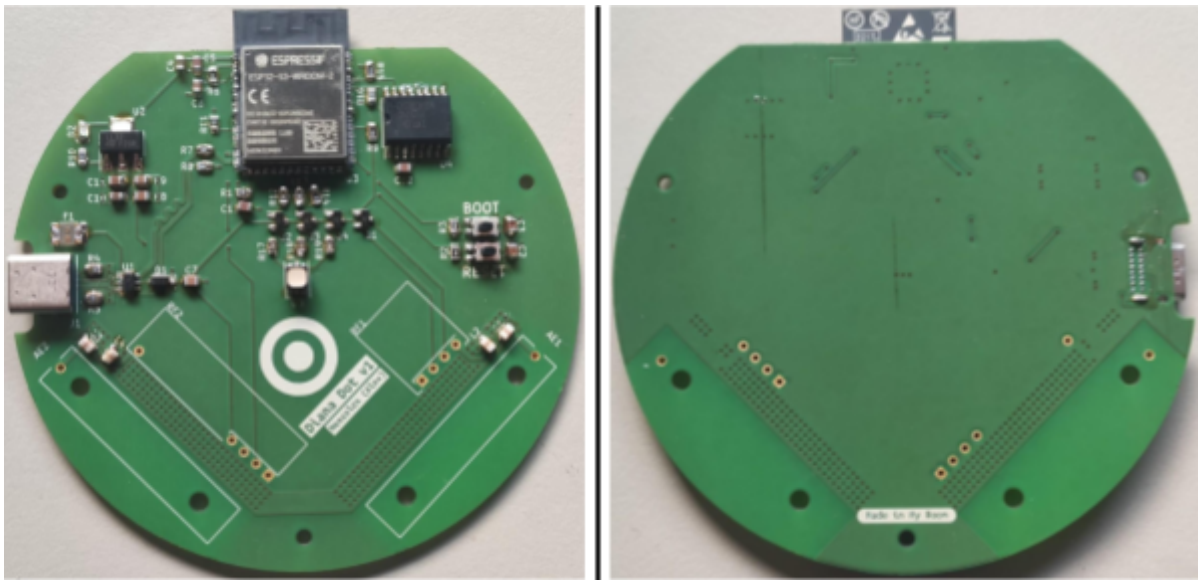


Figure 99: PCB with SMD components soldered top and bottom views.
Source: Compiled by the author.

THT components

The final step to manufacture the board is to solder the missing through-hole components. They simply consist of the two 433MHz modules and antennas. A soldering iron was used to do so. In the following *Figure 100* there is the result of the PCB with the THT components soldered. Notice that *Figure 107* contains the final result of the PCB with some tweaks done to make the board fully function. These problems are explained later on in its own section.

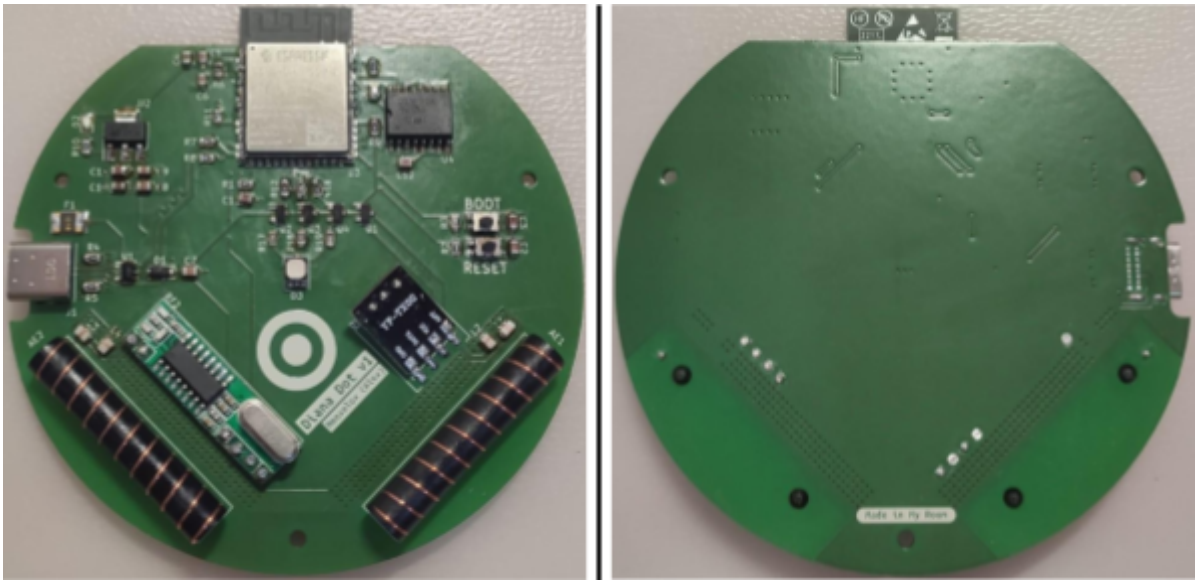


Figure 100: PCB with THT components soldered top and bottom views.
Source: Compiled by the author.

11.3. Testing

In order to test the designed PCB before passing onto its manufacturing, and as mentioned in the *objectives* section of this Thesis, the design rule checker (DRC) of the used CAD program was required to pass flawlessly configured with the manufacturer's parameters. The following *Figure 101* depicts so.

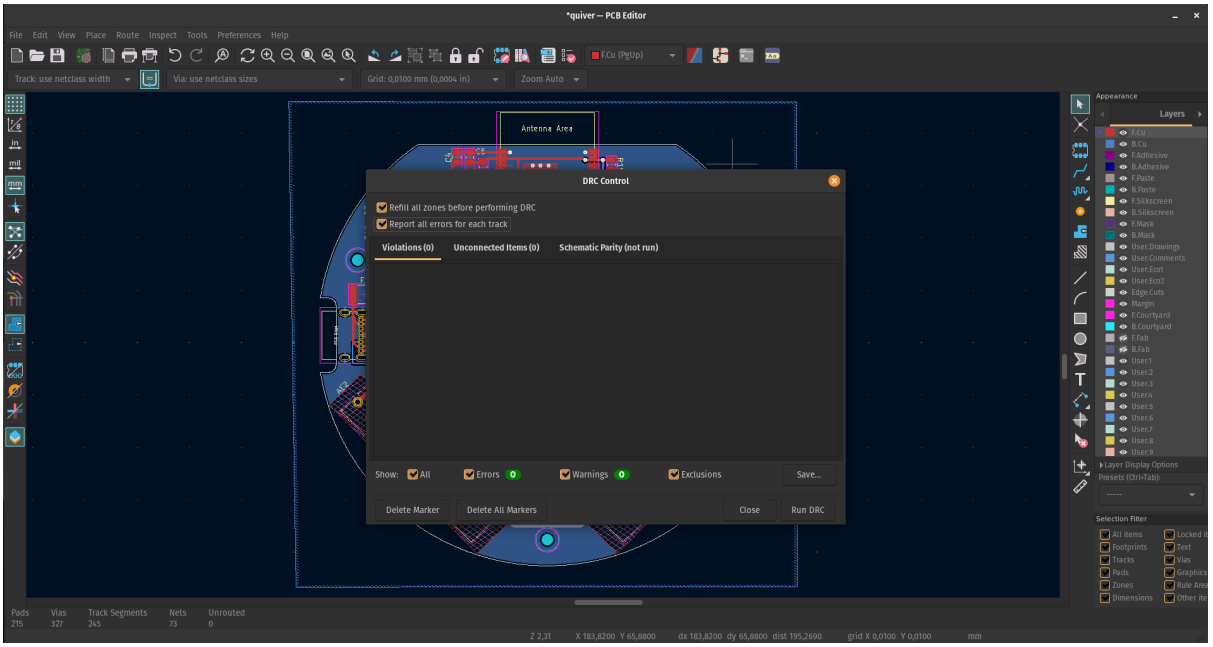


Figure 101: Design rule check of the PCB. *Source: Compiled by the author*.

When the manufactured PCB arrived it was obviously flashed with the version of the firmware at that time and the different features were tested thoroughly. Moreover, with the help of an oscilloscope, the different parts of the PCB were analyzed to check if they were functioning as intended, as seen below in *Figure 102*.

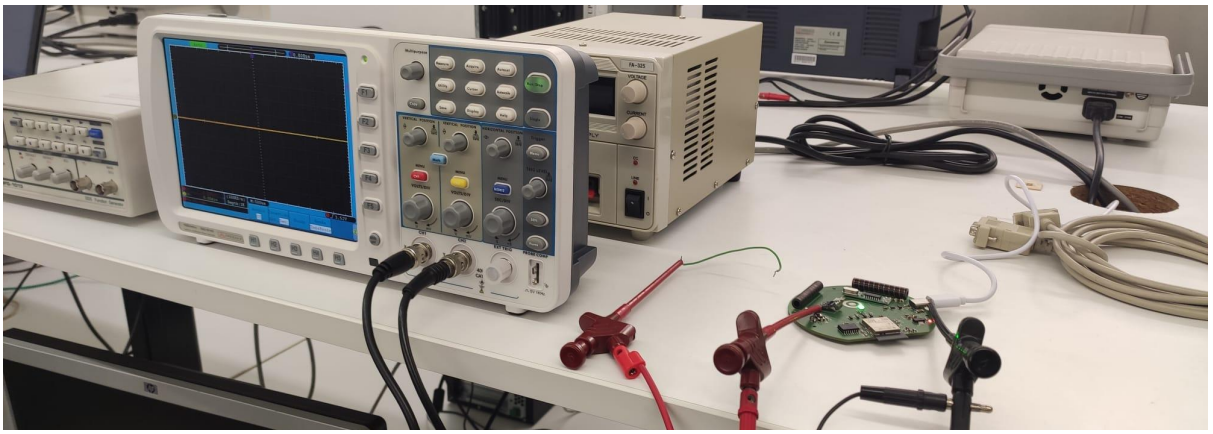


Figure 102: Testing of the PCB. *Source: Compiled by the author*.

11.4. Problems and solutions

As in the *schematic problems* section, some problems were detected after testing the manufactured PCB.

Transmitter is reversed.

The transmitter module had to be soldered reversed, as shown below in *Figure 103*, because the pin layout was reversed.

Figure 103: PCB part of the “reversed transmitter” problem. *Source: Compiled by the author*.

This simply happened because the footprint was wrongly designed the other way around. Luckily, the severity of this issue was nonexistent as all pins of the transmitter module are the same, so the solution taken was just to solder the module reversed.

Transmission antenna does not send anything.

This is the second cause of the transmission problem already mentioned in the *schematic problems* section. The transmission antenna still was not sending anything even after removing the offender inductor.

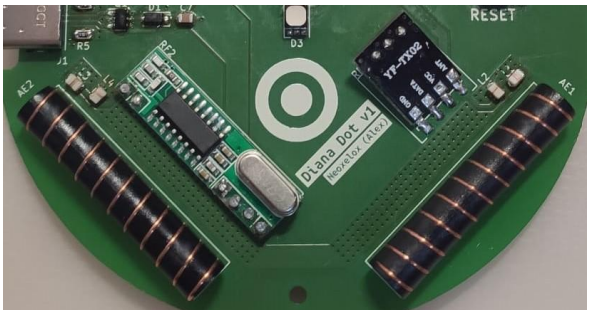


Figure 104: PCB part of the “transmission antenna does not send anything.” problem. *Source: Compiled by the author*.

After a lot of testing, I discovered that by putting a layer of an antistatic bag covering the antenna it works. The problem could supposedly be that there is not enough space between the two antennas and severe interference occurs because of antenna coupling. As said, the antistatic solution can be seen on *Figure 107* which contains the final result of the PCB.

11.5. Results

The final complete PCB design is included below in *Figure 105*.

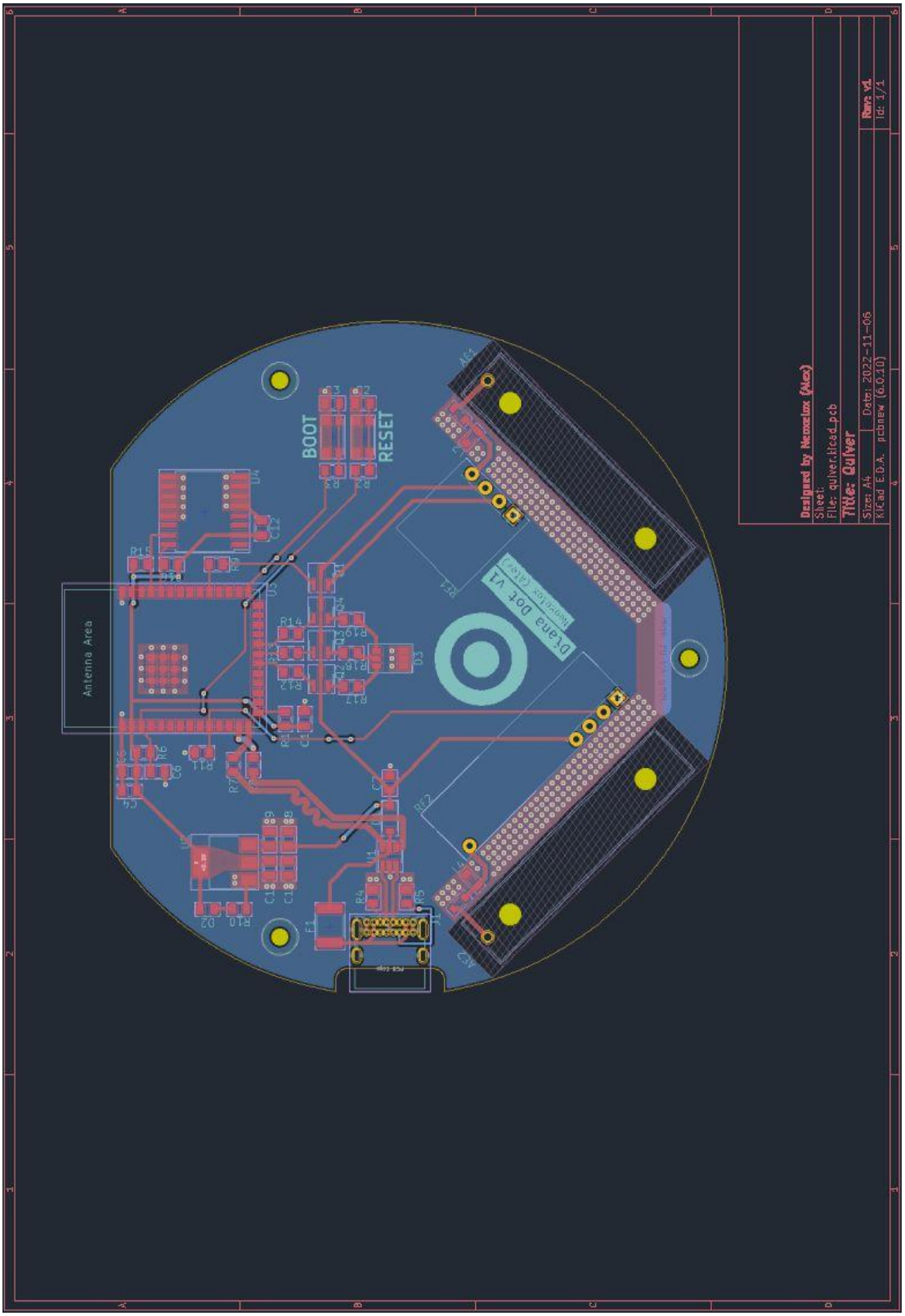


Figure 105: Complete PCB design. *Source: Compiled by the author*.

A large version of the 3D render of the PCB is included below in *Figure 106*.

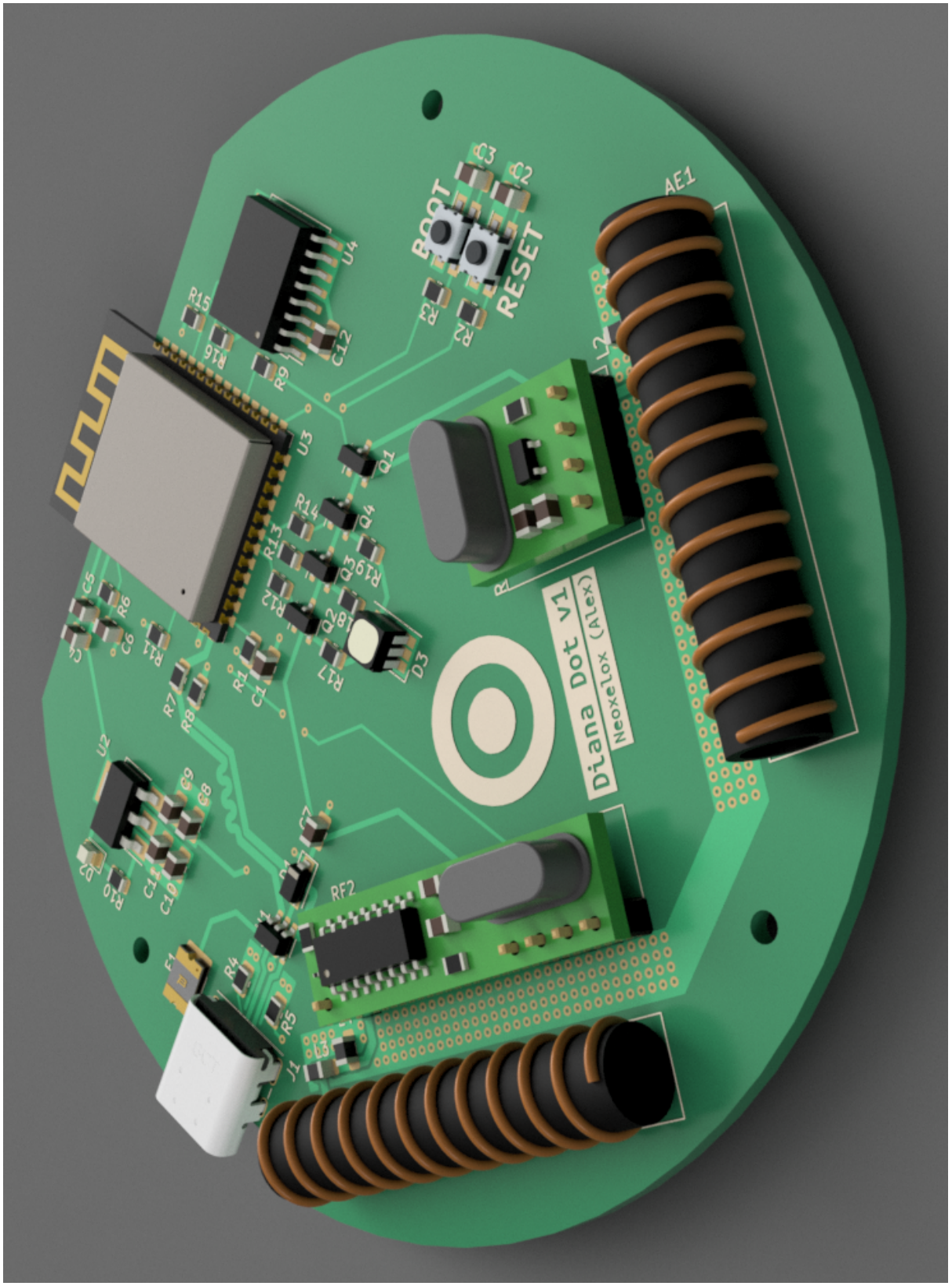


Figure 106: Complete PCB 3D render. *Source: Compiled by the author*.

The final complete manufactured PCB is included below in *Figure 107*.

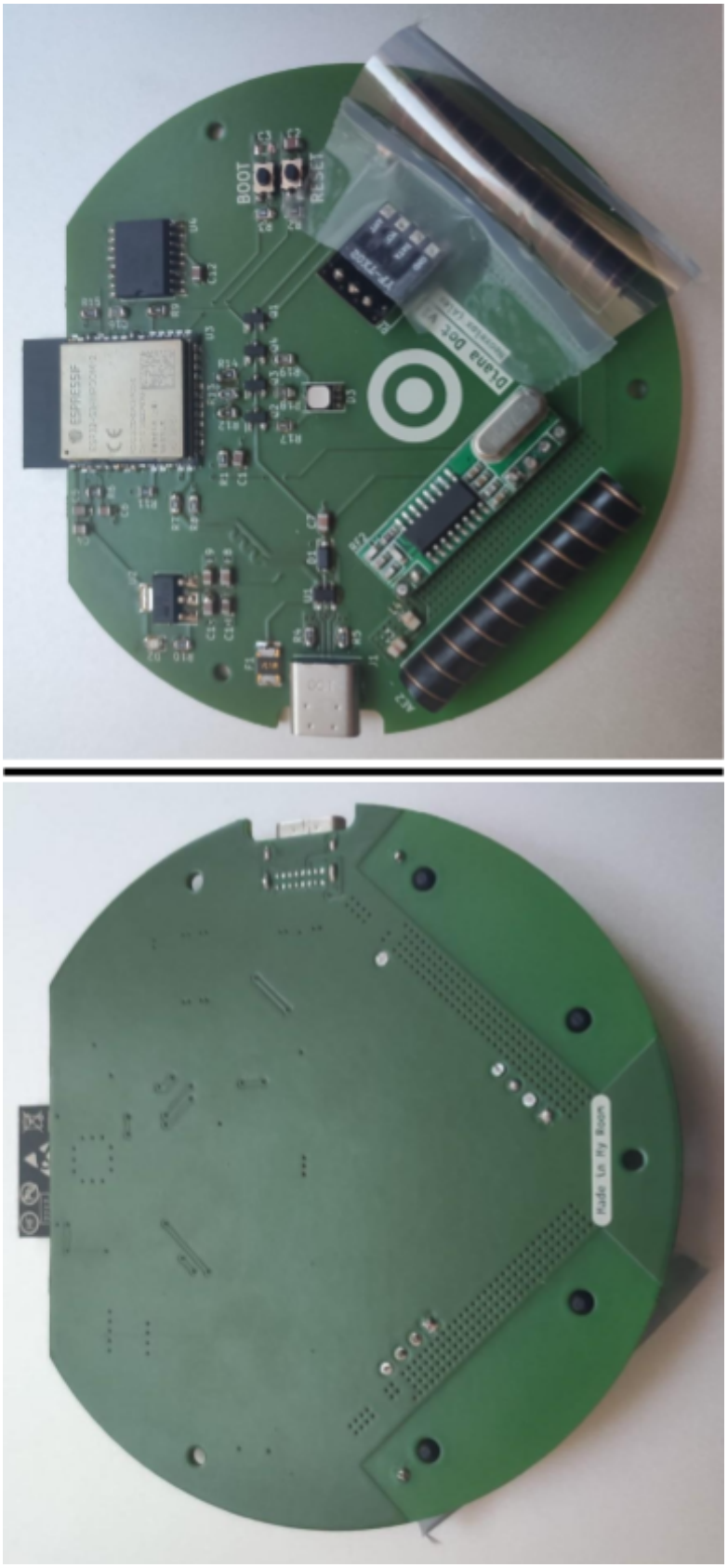


Figure 107: Complete PCB manufactured. *Source: Compiled by the author*.

12. Casing

12.1. Design

Body

The following *Figure 108* illustrates the body part of the casing.

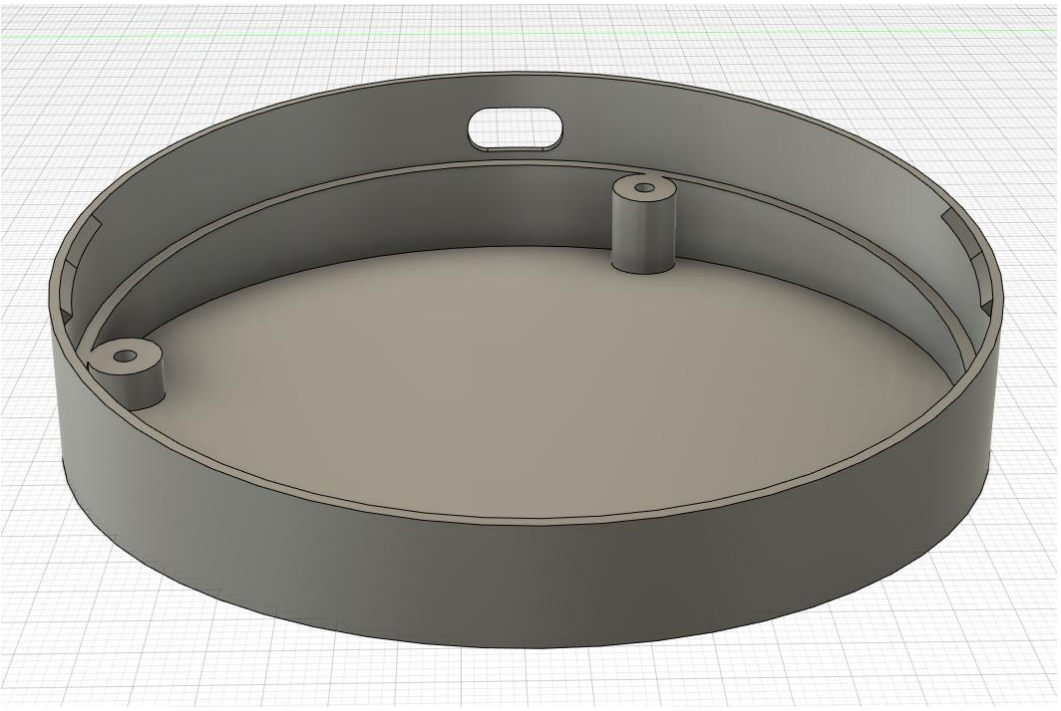


Figure 108: Body casing part. *Source: Compiled by the author*.

The body of the casing is made out of a smooth cylinder that was cut in angle with an horizontal plane, specifically at 7.5º. As already mentioned, this tilt (general tilt of keyboards) is made to make the status RGB LED more visible from the frontal side when the casing is placed in a high surface. Moving on to the interior, there is a central hole at the back. Its purpose is to expose the PCB's USB-C receptacle to the exterior so a cable can be connected through it. The hole has been designed with bigger proportions to have tolerance.

As the PCB will be installed inside the casing and secured with three M2 screws, holes were placed in the model (one is hidden by the frontal face) at their same position as in the designed PCB. Their altitude varies following the 7.5º tilt so that the PCB is installed in a position that lets the lid and the body close loosely. Moreover, a little ledge has been designed in order for the PCB to rest. The wall thickness is just enough to make the casing robust, but not too much in order not to waste time and printing material as the casing won't be moved.

The two symmetric extrusions on both sides of the body are for the snap-fit mechanism. They do not cover all the circumference of the body as it would make the case very hard to open.

Lid

The following *Figure 109* illustrates the lid part of the casing.

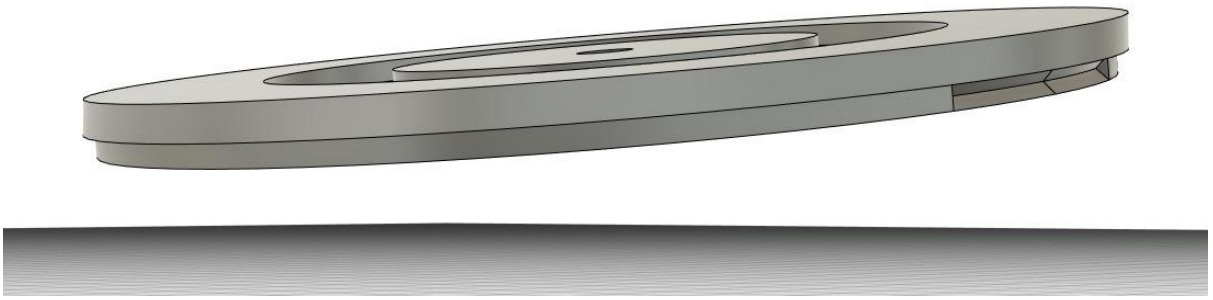


Figure 109: Lid casing part. *Source: Compiled by the author*.

The lid of the casing is made from the same cylinder that made the body, cutting it with the same tilted plane at the same angle but higher up. As decided, the casing must resemble a bulls-eye, so a circumference has been negatively extruded downwards as the manufacturing mechanism used for the casing, 3D printing, does not allow for two colors. Moreover, it has a central hole to let the light out from the RGB LED inside. On that hole a little diffusive plastic lens will be placed so that the light is concentrated in one place. The indentations for the snap fit closing mechanism have been placed at the same positions as the extrusions of the body.

Snap fit closing

The following *Figure 110* illustrates the snap fit closing part of the casing.

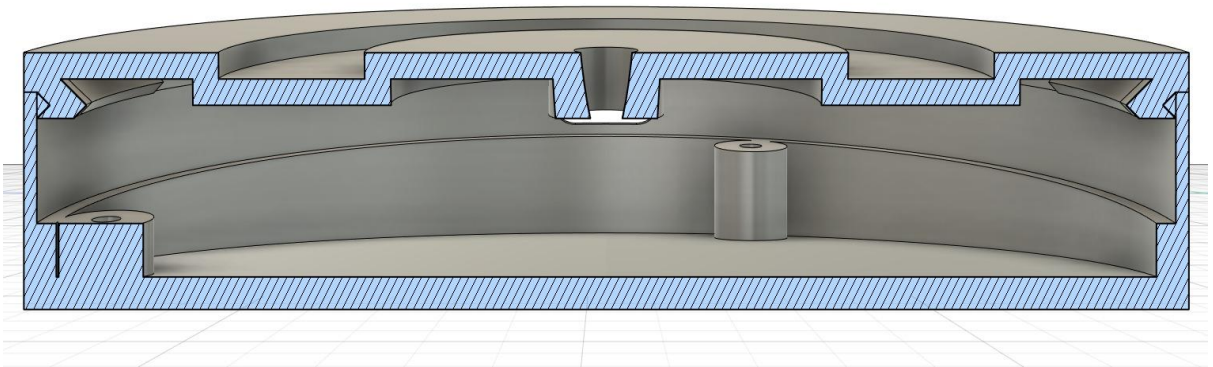


Figure 110: Snap fit closing casing part. *Source: Compiled by the author*.

The cross section analysis seen in the figure shows the snap fit mechanism, when the lid is closed onto the body, on both sides. When the casing closes the extrusions from the body attach to the indentations on the lid. The printing material used, PLA, is flexible enough to let the indentations bend a little to surpass the extrusions and then attach to them firmly.

3D preview

As stated in the *requirements* section of this Thesis, a 3D render of the casing should be included. It allows to preview how the final manufactured casing will look like and although not shown in the following figures, it also allows to check if the PCB will fit inside the casing or not. Such render is included later on in the *casing results* section.

The following *Figure 111* depicts the 3D render of the casing from the top and the bottom.

Figure 111: Casing 3D render top and bottom views. *Source: Compiled by the author*.

Figure 112 below, depicts the 3D render of the casing from an angular perspective in order to see the depth of it.

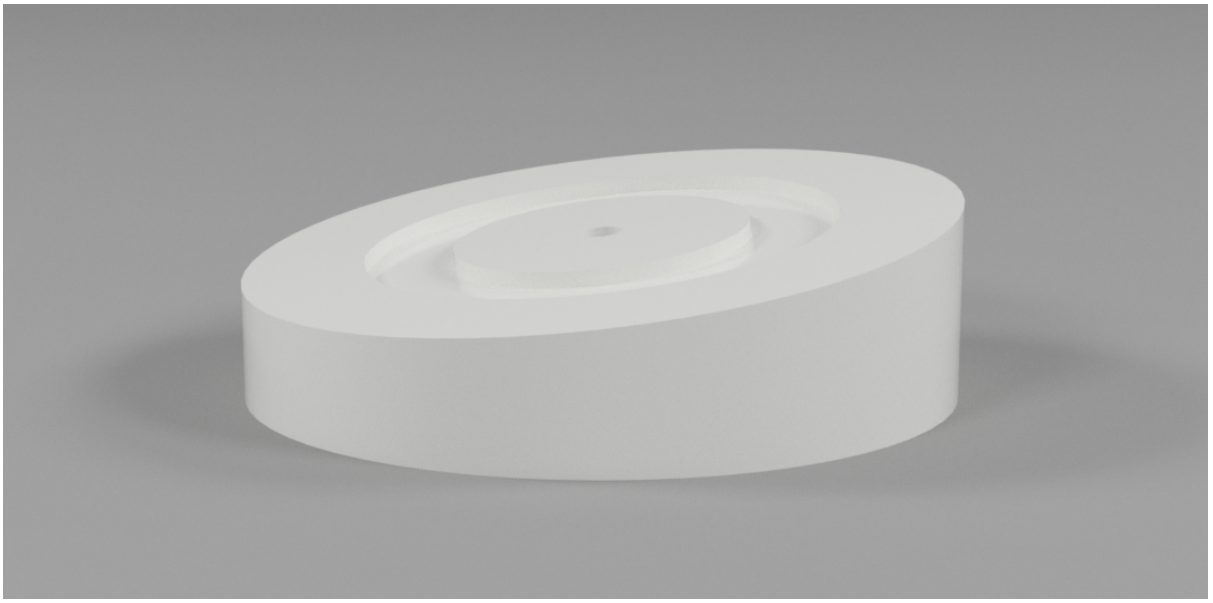


Figure 112: Casing 3D render angle view. *Source: Compiled by the author*.

12.2. Manufacturing

Although the final "good" version of the casing has been manufactured by a local 3D-printing facility to ensure that a smooth result is achieved, a quick first version of the case, without the snap fit closing mechanism, was printed previously by myself using my own 3D printer. This allowed me to verify that the design was correct before ordering the professional printing.

This section is the process of such printing as photos couldn't be taken in the local 3D-printing facility.

Slicing

The previous step necessary to be able to 3D-print any model is to generate the instructions (G-code) that the 3D printer will follow in order to deposit each layer of plastic and little by little form the designed model. This process is automatic and can be done through any 3D "slicing" program. In this case the used one was Ultimaker Cura, as seen below in *Figure 113*.

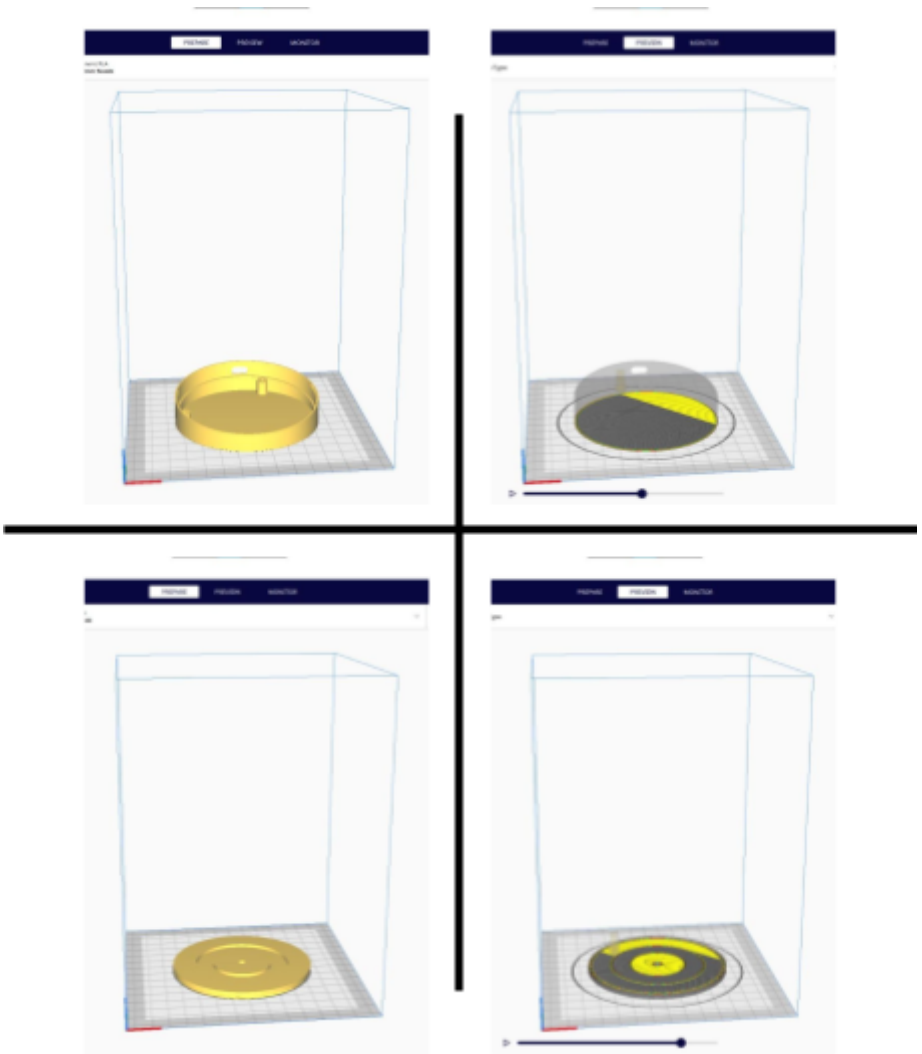


Figure 113: Casing slicing. *Source: Compiled by the author*.

Printing

The following step is to upload such instructions to the 3D printer and start printing.

As mentioned before, the printing shown in this section is from myself. It must be noticed that I only had black PLA on hand.

However, as decided before in the *taken approach* the real material selected for the construction of this casing is a white semi-translucent PLA, which will allow easy manufacturing while maintaining strong and resistive properties. The white color is selected to have a good contrast with the light of the RGB LED, and making it translucent allows the light to better propagate and blur.

The following *Figures 114 and 115* depict the 3D printing process of both body and lid parts of the casing.

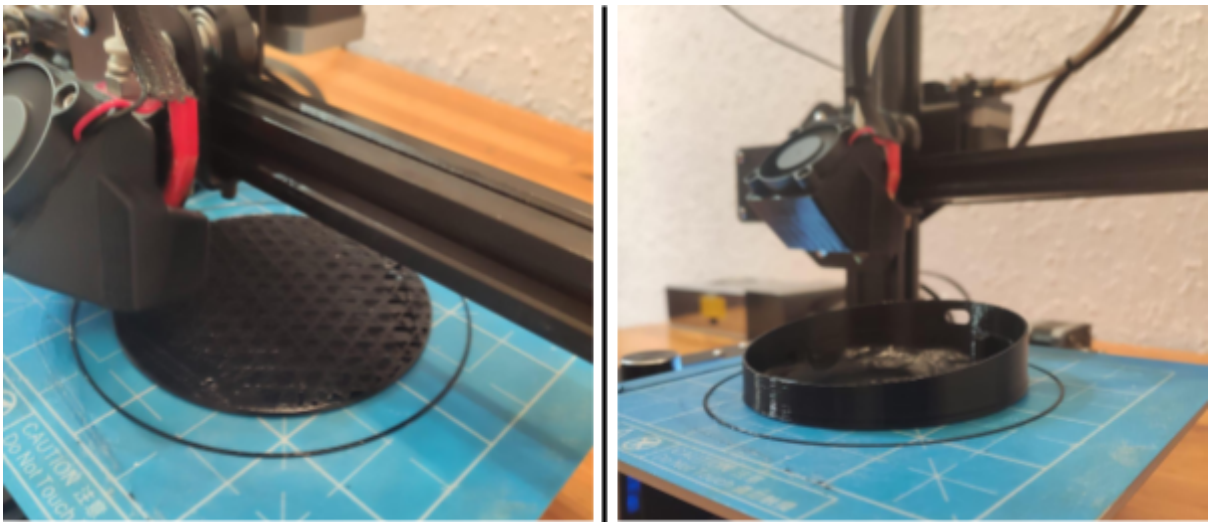


Figure 114: Casing body printing. *Source: Compiled by the author*.

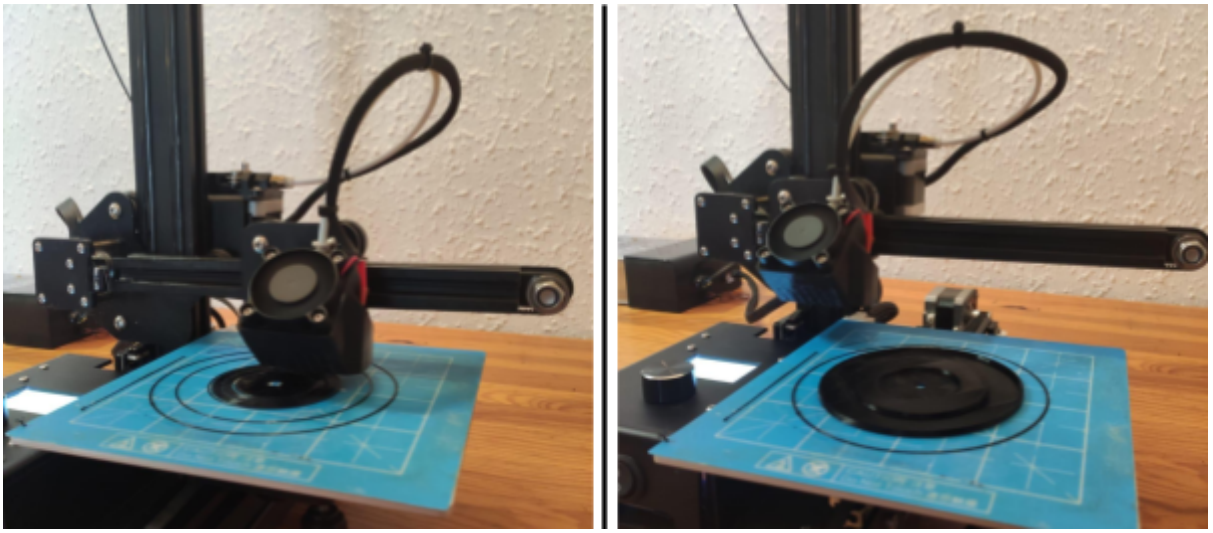


Figure 115: Casing lid printing. *Source: Compiled by the author*.

Below in *Figure 116* there are the final 3D printed casing body and lid from the local 3D-printing facility. As said before, the decided material was white PLA. Its semi-translucent capabilities are shown later on in the *casing results* section.

Figure 116: Casing body and lid manufactured. *Source: Compiled by the author*.

Finishing

Finally, the little diffusive plastic lens that will be placed on the lid's hole so that the light is concentrated in that place, was made from a semi-translucent plastic clothes hanger. The hook part of it was cut in half and then polished as seen in the following *Figure 117*.

Moreover, on the right there is an example of what it looks like after placing it on top of the RGB LED of the PCB.

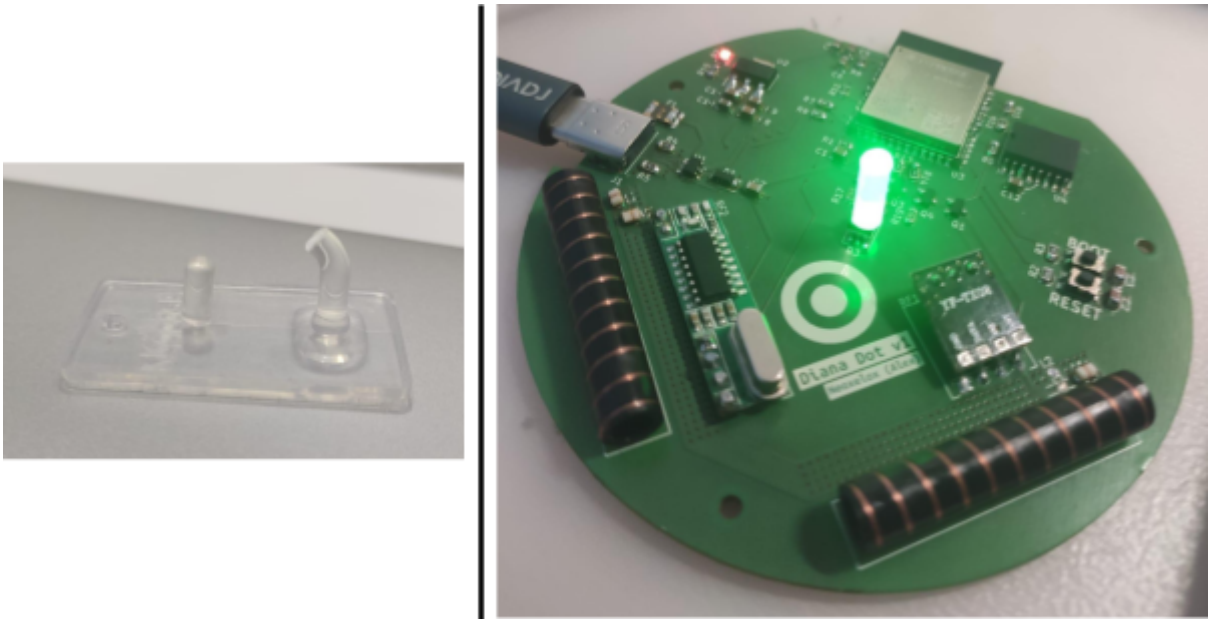


Figure 117: Casing RGB LED plastic lens. *Source: Compiled by the author*.

12.3. Testing

In order to test the designed casing before passing onto its manufacturing, the previously created 3D model of the PCB was used to perform a simulation to check whether it fitted, as seen in the following *Figure 118*. This helped to ensure that the case would be manufactured without any measurement or mechanical errors.

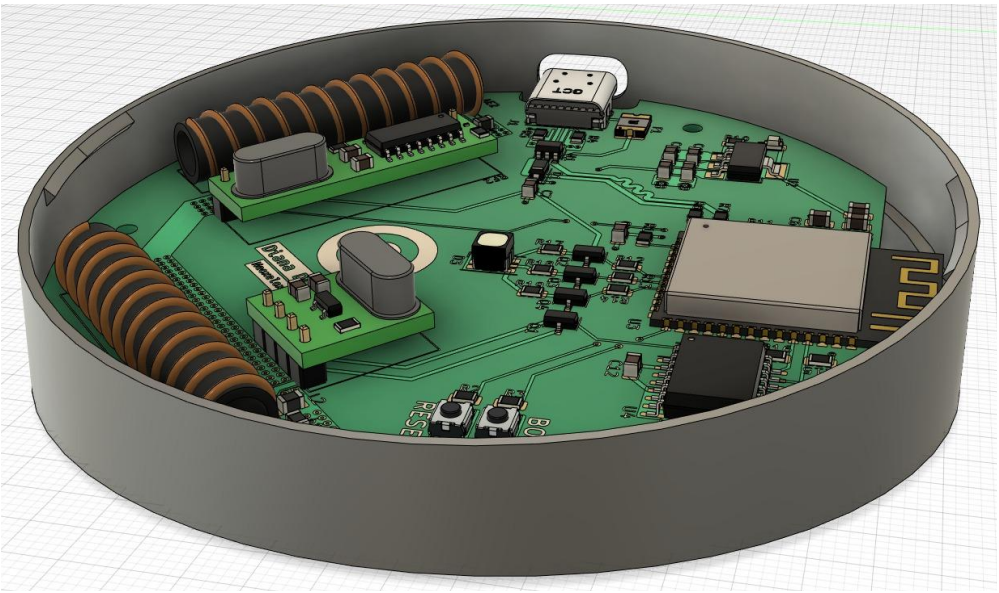


Figure 118: Casing and PCB simulation. *Source: Compiled by the author*.

When the final version of the casing was manufactured by the local 3D printing facility, a test to check whether the real PCB actually fitted was performed as seen below in *Figure 119*.

Figure 119: Testing of the casing. *Source: Compiled by the author*.

12.4. Problems and solutions

As in the others *problems* section, some problems were detected after testing the manufactured casing. However, almost most of them were visual issues due to the limitation in resolution of the 3D printing mechanism.

Snap fit closing is too tight.
Closing the case with the snap fit mechanism, seen below in *Figure 120*, is sometimes a little bit difficult.

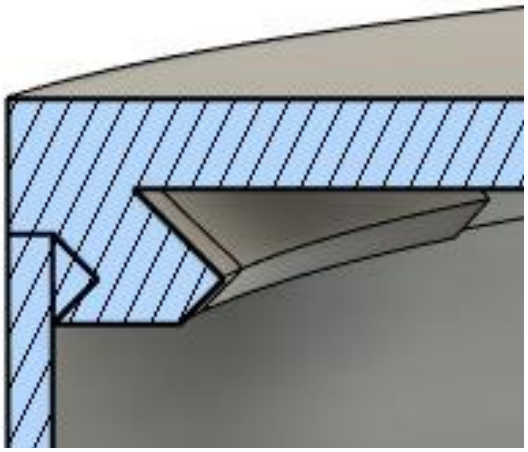


Figure 120: Casing part of the "snap fit closing is too tight" problem.
Source: Compiled by the author.

This is simply due to the little to no tolerance of the designed indentations. As seen in the figure, both the extrusion and the indentation match perfectly. However, if there was some tolerance the attaching process would have been less difficult.

This problem does not present any severity as the case can be closed, so no action was taken.

12.5. Results

The final complete casing design is included below in *Figure 121* along with its dimensions.

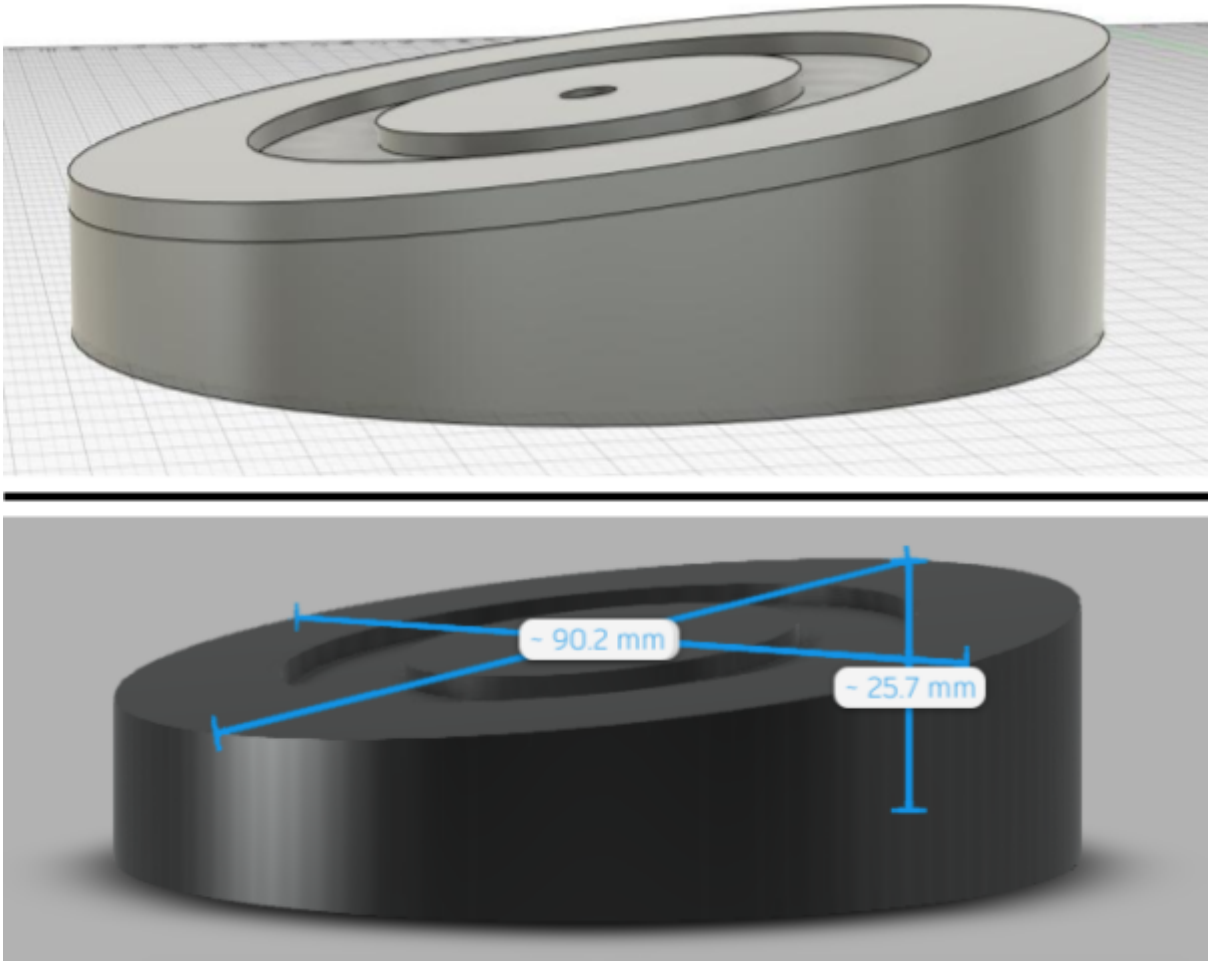


Figure 121: Complete casing design. *Source: Compiled by the author*.

A large version of the 3D render of the casing is included below in *Figure 122*. It also features the previous created 3D model of the PCB in order to give perspective of the two together.

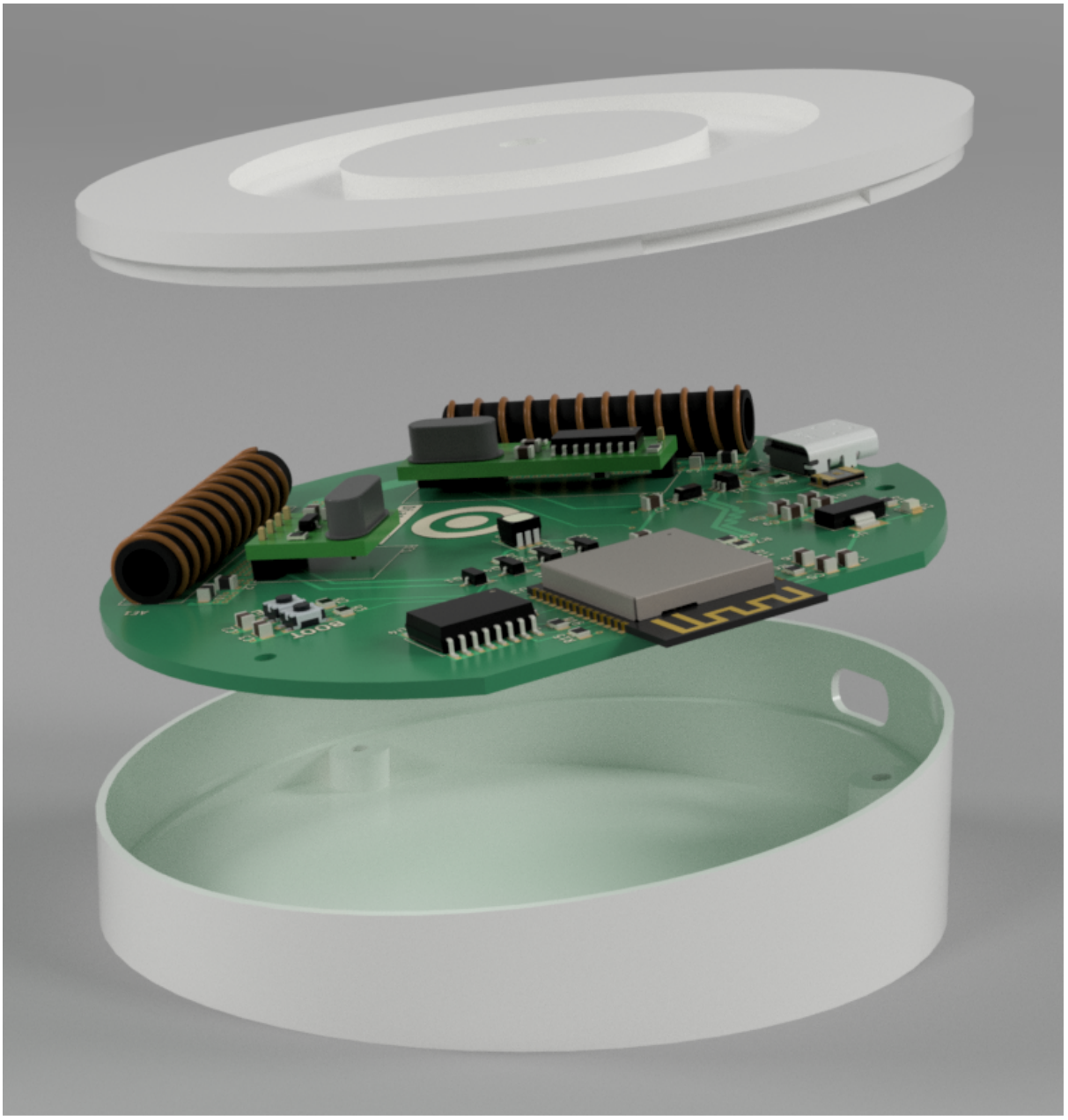


Figure 122: Complete casing 3D render. *Source: Compiled by the author*.

The final complete manufactured casing is included below in *Figure 123*. The casing is presented in both light and dark conditions in order to see the semi-translucent capabilities of the chosen printing material. The noticeable red light, coming from a side of the casing on the dark version of the figure, is the power LED indicator included in the PCB to let the user know the device is switched on. Although it could seem that in light conditions such LED is not noticeable, looking closer at the casing reveals the LED.

Figure 123: Complete casing manufactured. *Source: Compiled by the author*.

13. Packaging

13.1. Design

Dieline

The manufacturer used to produce the packaging samples, Packly, already provided me with the base dieline for the box type I selected, “hanging box with crash lock bottom”, as seen below in *Figure 124*.

A dieline is the template needed to ensure the correct layout of the final physical package. This template serves as a diagram that marks all the folds and cut lines of the package in a flattened form.

With this diagram, the graphical design was carried on in Figma. The complete dieline is included in the *results* section.

The red lines on the dieline mark the folds of the packaging, the blue ones mark the separation of each face and the green ones are “bleed lines”. The background color of a face must reach those lines as they serve as a mechanism to ensure the color of the face stretches out all the way to the folds so there are no white lines between the separation of each face if the colors are different from one to the other.

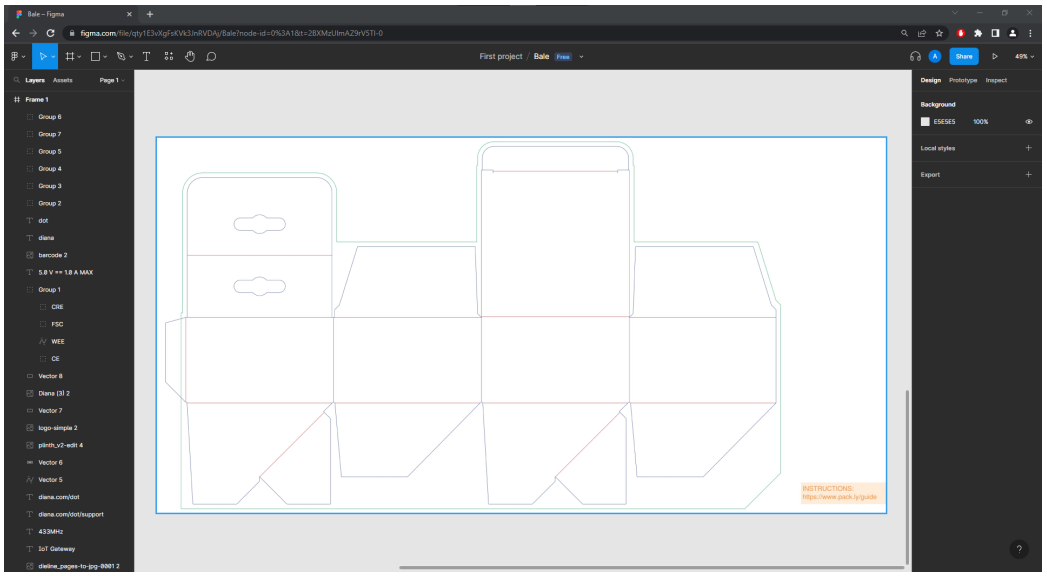


Figure 124: Packaging dieline. *Source: Compiled by the author*.

Top

The following *Figure 125* illustrates the top part of the packaging. As mentioned in the *taken approach* section the top face will be the most important as it is the first one to make eye contact with the hypothetical consumer.

Therefore, it will include the name of the product "diana dot" with a descriptive tagline "433MHz IoT Gateway", a little logo of the company and a 3D render of the product that's inside, so the consumer can have an idea of what to expect. Moreover, the indigo color above the face is the colored flap that will serve as the closing mechanism for the box.

Figure 125: Top packaging part. *Source: Compiled by the author*.

The presented 3D render comes from the 3D model of the casing and has been post-processed a little using the cinema-grade software Blender, as seen below in *Figure 126*.

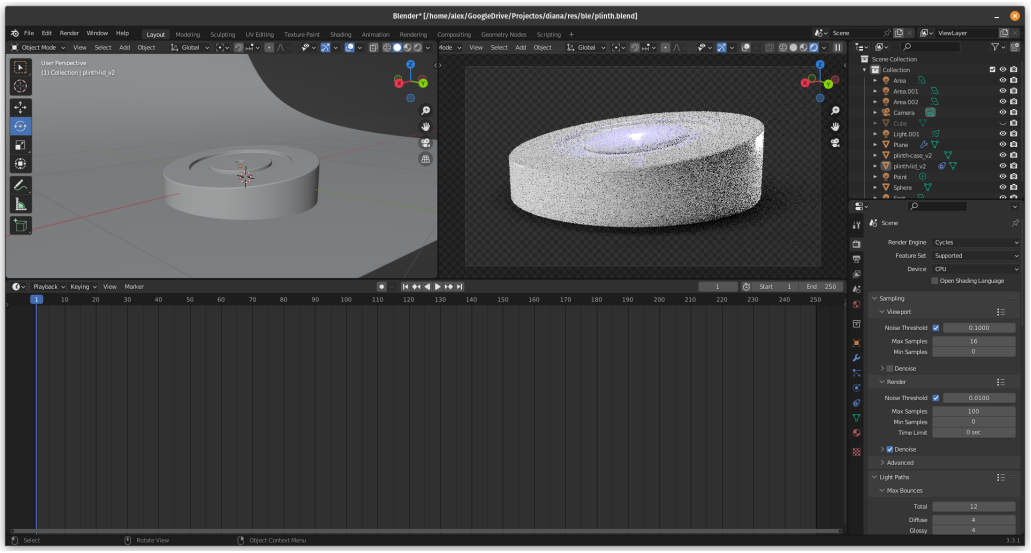


Figure 126: Post-processing of the 3D model of the casing. *Source: Compiled by the author*.

Bottom

The following *Figure 127* illustrates the bottom part of the packaging. As it can be seen it does not include any graphical design whatsoever, only a little watermark, as it is the least important part of the package visually. However, it features the folding crash-lock mechanism that allows the package to be folded and keeps it mounted.

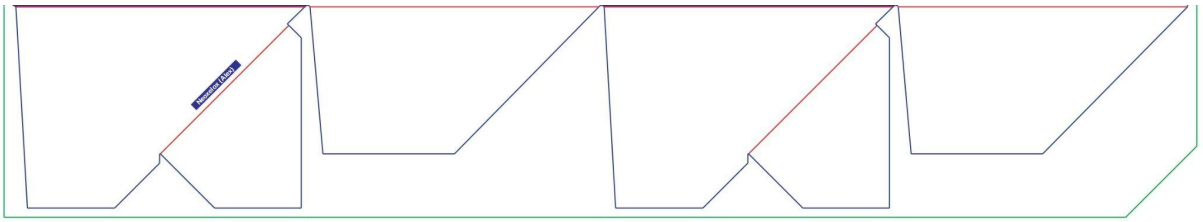


Figure 127: Bottom packaging part. *Source: Compiled by the author*.

Front

The following *Figure 128* illustrates the front part of the packaging. Includes a big version of the hypothetical company logo, which is a bulls-eye.

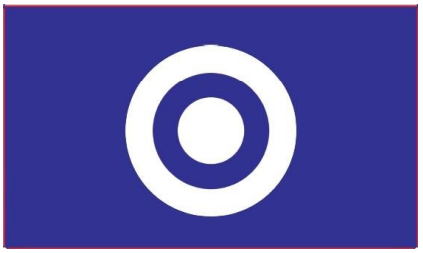


Figure 128: Front packaging part. *Source: Compiled by the author*.

Back

The following *Figure 129* illustrates the back part of the packaging (rotated so it fits). It includes some electrical specifications "5V 1A MAX", stamps and a (fake) barcode with its hypothetical assigned UPC tracking code. Although it is hard to see in the figure, both of the upper faces form the back hanging hole. It can be seen in the final manufactured package.

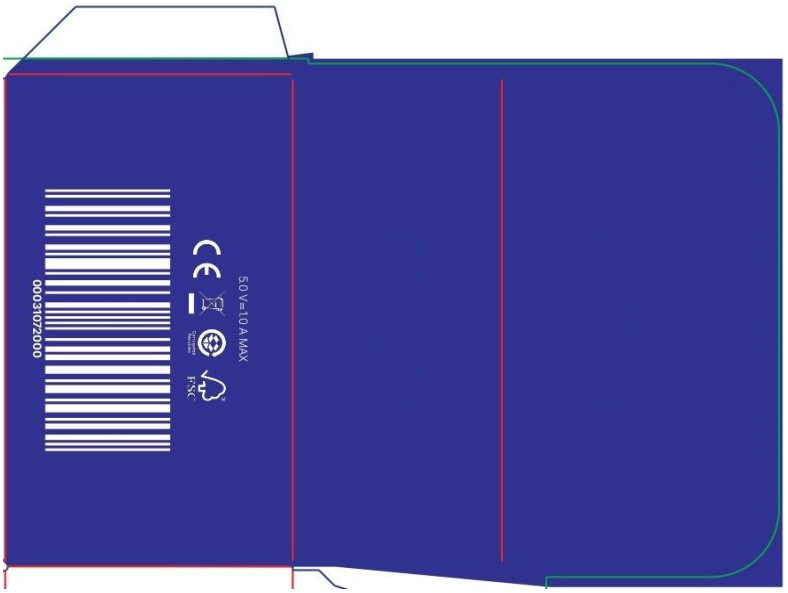


Figure 129: Back packaging part. *Source: Compiled by the author*.

Left

The following *Figure 130* illustrates the left part of the packaging. As decided, the side faces include product information.

The left one serves as a quick guide on how to set up the device for the first time. Instructions are short and direct so everyone can understand swiftly what to do.

Moreover, a link (fake) to an extended support guide is also included on the bottom right corner.

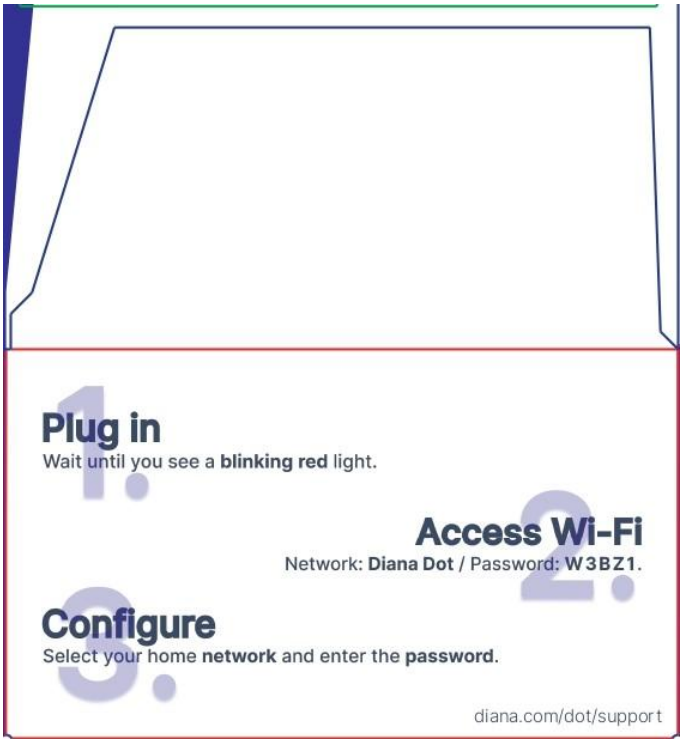


Figure 130: Left packaging part.
Source: Compiled by the author.

Right

The following *Figure 131* illustrates the right part of the packaging.

In this case, this face serves as a showcase of the different features of the device with little descriptions. The functionalities included are the most important ones: the band where the device works at, what the user can do and the extra scheduling function.

Moreover, a link (fake) to the product's landing page is also included on the bottom right corner.

Figure 131: Right packaging part.
Source: Compiled by the author.

3D preview

As stated in the *requirements* section of this Thesis, a 3D render of the packaging should be included. It allows to preview how the final manufactured package will look like and also to check if the graphical designs will fit the packaging faces or not.

The following *Figure 132* depicts the 3D render of the packing from the left and the right sides. The top lid is opened a little to let glimpse the depth of it.

Figure 132: Packaging 3D render left and right side views. *Source: Compiled by the author*.

Figure 133 below, depicts the 3D render of the casing from the back. The hanging hole is visible.

Figure 133: Packaging 3D render back view. *Source: Compiled by the author*.

13.2. Manufacturing

Package

The first step taken to manufacture the packaging was to reach Packly to print the designed dieline. It can be done through their website. The model, dimensions and material of the wanted package must be selected as well as uploading the dieline in PDF, as it can be seen below in *Figure 134*.

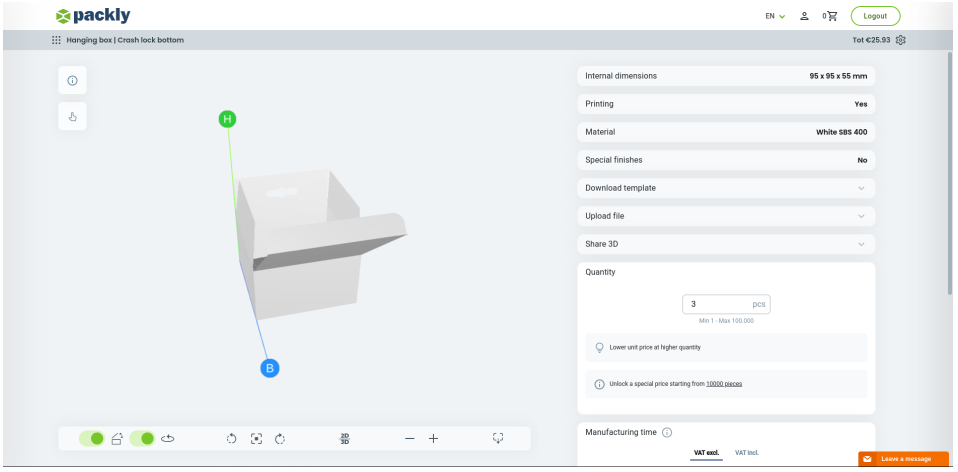


Figure 134: Packly packaging order overview. *Source: Compiled by the author*.

The material selected for the construction of this package is a white coated 400gsm cardboard, which will allow easy manufacturing while maintaining sufficient strong and resistive properties.The graphics designs of the dieline will be printed onto the coated cardboard which will make the outcome very professional.

As the selected vendor is from the European Union, order to arrival took less than 2 weeks. The different manufactured packaging samples came fully unfolded as shown below in *Figure 135*.

Figure 135: Packly packaging manufactured. *Source: Compiled by the author*.

Thanks to the selected package model featuring a crash-lock mechanism in the bottom, mounting a package is quick and simple, just giving a little bit of form to the box and the mechanism automatically locks in place, as shown in the following *Figure 136*.

Figure 136: Packaging bottom mount. *Source: Compiled by the author*.

The last step is to remove the placeholders from the hanging hole and fold it over, as shown below in *Figure 137* (rotated so it fits).

Figure 137: Packaging hanging holes. *Source: Compiled by the author*.

Separator

As mentioned in the *taken approach* section, a polystyrene sheet is included in the packaging separating the proper device with a USB-C cable and charger, which are also included, as it can be seen in the following *Figure 138*.

Figure 138: Packaging polystyrene separator. *Source: Compiled by the author*.

13.3. Testing

In order to test the designed packaging before passing onto its manufacturing, the 3D preview capability of Packly was used to simulate the result and check whether the designed dieline correctly fitted all faces, as seen in the following *Figure 139*.

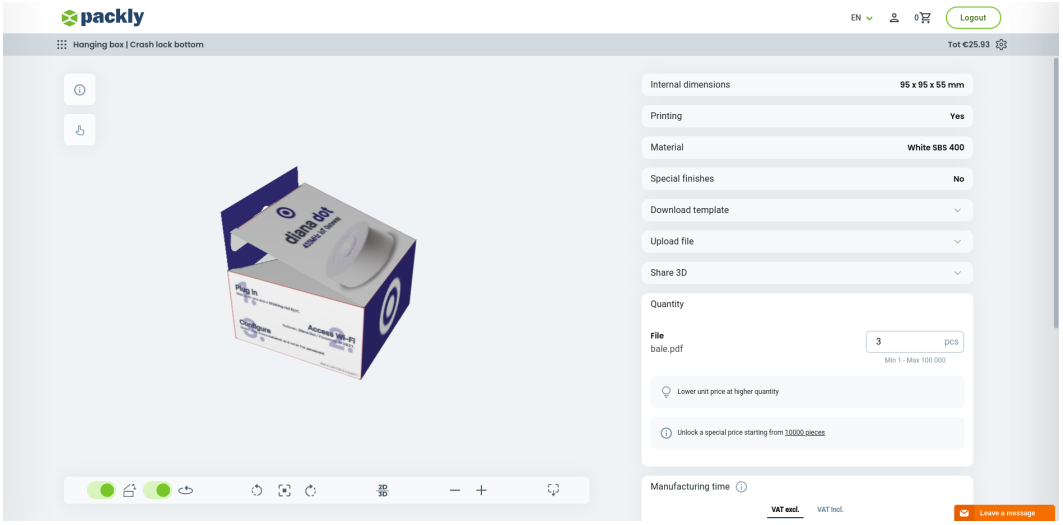


Figure 139: Packaging dieline simulation. *Source: Compiled by the author*.

When the manufactured packaging arrived, a test was performed to check whether the casing, the polystyrene separator and the included USB-C cable and charger actually fitted inside, as seen below in *Figure 140*.

Figure 140: Testing of the packaging. *Source: Compiled by the author*.

13.4. Problems and solutions

Unlike the others *problems* section, no problems were detected after testing the manufactured packaging. However an incongruence in the second step of the designed setup instruction was eventually found, as seen in the following *Figure 141*.

Figure 141: Packaging setup instructions incongruence.
Source: Compiled by the author.

The included password of the softAP network, that the device itself creates, on the packaging has only 5 characters, as it was thought to be as simple and quick as possible to reduce friction in the onboarding process. The issue is that the network is created with a WPA2 security scheme as already mentioned in the *taken approach* section, and that and that forces the password to be of 8 or more characters, so the included password is invalid.

This incongruence happened because as mentioned before in the *revisited gantt chart* section, most of the firmware tasks were postponed until after the creation of the schematic, pcb, casing and packaging, to be able to finish all things on time.

This does not present any severity as it is a visual issue with zero real importance, so no action was taken.

13.5. Results

The final complete packaging design is included below in *Figure 142*.

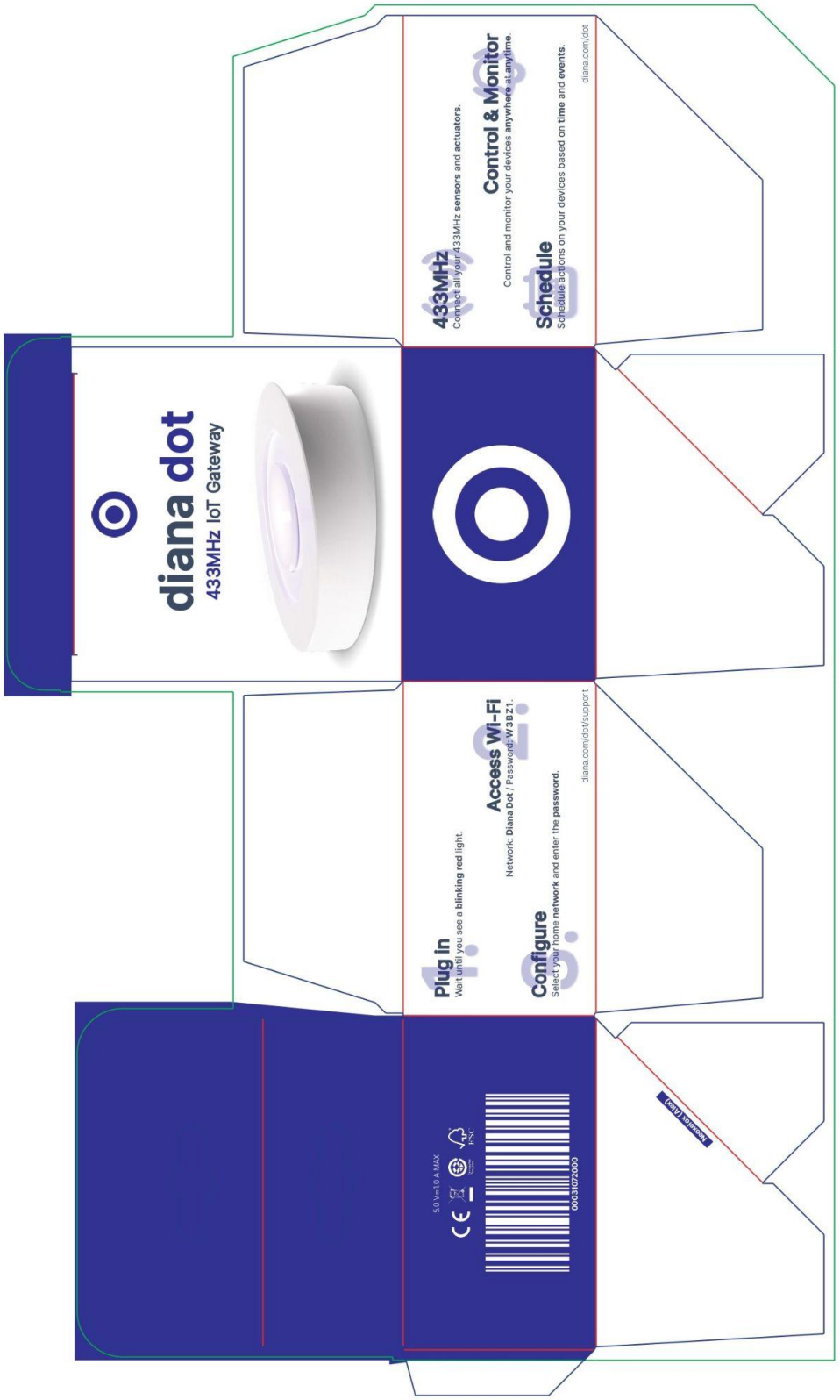


Figure 142: Complete packaging design. *Source: Compiled by the author*.

A large version of the 3D render of the packaging opened is included below in *Figure 143*.

Figure 143: Complete packaging 3D render. *Source: Compiled by the author*.

The final complete manufactured packaging is included below in *Figure 144*.

Figure 144: Complete packaging manufactured. *Source: Compiled by the author*.

14. Firmware

The codebases for both backend and frontend are very big (+20k lines of code) and have several complex parts. In this section the implementation of the most important bits of both projects is explained and summarized through flowcharts and tables. The reasoning of why and what every software part does was explained previously in the *taken approach* section.

The repositories for both *bow* [67] (backend) and *arrow* [68] (frontend) are included as links in the bibliography. Both are open source with a GPL-3 licensing.

However, on the backend side there were two objectives that couldn't be reached (more on that in the *future work* section). On the one hand, the OTA updates feature was very complex, time consuming and zero rewarding. The firmware can be perfectly updated through USB-C via a JTAG interface. On the other hand, all TLS layers, that is SSL on the HTTP server and client, couldn't be done as they were too resource intensive for the microcontroller, and the gains of them did not overcome the losses.

Moreover, although the backend does indeed serve a frontend by itself, as required by the objectives, the interface of it has not been fully finished as there was no time left to do so. Therefore, this section will not cover anything more in detail about the frontend.

14.1. Design

Architecture

As said before in the *taken approach* section, the backend firmware was built using the ESP-IDF toolchain from the microcontroller's vendor Espressif. *Table 11* below summarizes the project structure for the backend project with the main directories and files.

NAME		COMMENT
BUILD		Contains the intermediate C object files and the final firmware binary
COMPONENTS		Contains the source code for the different software components
	CHRON	Contains the source code for its software component
	DATABASE	
	DEVICE	
	LOGGER	
	PROVISIONER	
	ROLE	
	SERVER	
	STATUS	
	TRIGGER	
	USER	
DOCS		Contains HTTP REST API documentation
MAIN		Contains the source code for the main entrypoint
SCRIPTS		Contains build, lint and monitoring scripts written in Python
STATIC		Contains the embedded frontend web application
	FRONTEND.GZ	

VENDOR		Contains the source code for the different third-party libraries used
	CAPDNS	Captive portal DNS server by Espressif
	CCRONEXPR	Cron expression parsing by Staticlibs
	DS3231	DS3231 driver by UncleRus
	I2CDEV	I2C driver by UncleRus
CMAKELISTS.TXT		CMake definitions for compiling and linking the binary
PARTITIONS.CSV		Partitions definitions for the microcontroller
SDKCONFIG		Configuration definition for the toolchain

Table 11: Backend firmware project structure. *Source: Compiled by the author*.

As the backend follows a dependency injection pattern through singletons, the following *Figure 145* depicts a dependency graph between the software components.

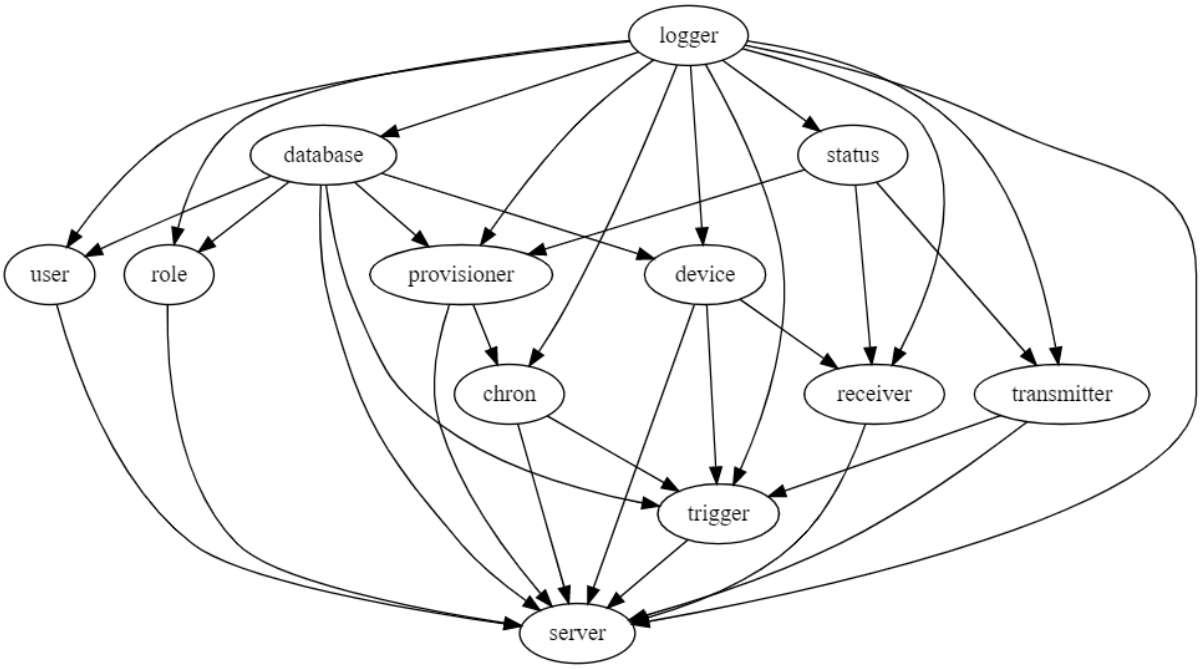


Figure 145: Backend firmware dependency graph. *Source: Compiled by the author*.

As said, the "main" directory contains the task launched by the toolchain itself when the microcontroller boots, which is the main entrypoint of the backend. As such, its only responsibility is to create all singletons in the above dependent order.

Device definitions

As said in the *initial research* section of the communication of the 433MHz devices, there is not a common protocol used. However, a generic algorithm can be implemented by parametrizing the expected phases for each protocol known and trying the best to match this model against the real received signal. The parameters of each protocol can be extracted from the previous results of the *data analysis* section of each test subject.

The data structure containing the protocol phases and parameters it's called "Protocol". The implementations is as follows:

```
class Protocol
{
public:
    struct Data
    {
        uint8_t Zero[MAX_DATA_ZERO_PULSES]; // Divisions
        uint8_t One[MAX_DATA_ONE_PULSES];   // Divisions
    };

    uint32_t Width;                         // Microseconds
    uint8_t Sync[MAX_SYNC_PULSES];          // Divisions
    uint8_t Preamble[MAX_PREAMBLE_PULSES]; // Divisions
    Data Data;
};
```

From now on, there will be no distinction between pulses and gaps, everything will be called a pulse, as both are bounded periods of time with the only difference that the pulse does transmit and the gap is just the opposite, does not transmit.

The width of a protocol defines the average measured width of a single pulse in microseconds, as it cannot be negative, it is implemented as a 32-bit unsigned integer.

The sync phase of a protocol is defined by the sequence of pulses that forms it. Pulses are 8-bit unsigned integers encoded as divisions of the base protocol width to save space. The maximum allowed length for this phase is 4 pulses, the 2 common at the start plus 2 extra pulses that appeared in some protocols.

The preamble phase of a protocol is also defined by the sequence of pulses that forms it. Pulses are also 8-bit unsigned integers encoded as divisions of the base protocol width to save space. The maximum allowed length for this phase is 24 pulses, which is the length of the wireless motorized blinds protocol's preamble phase (the only protocol that included preamble).

Finally the data phase of a protocol is defined by the sequence of pulses that forms a bit, either a logic zero or a logic one. Pulses are also 8-bit unsigned integers encoded as divisions of the base protocol width to save space. The maximum allowed length for each bit is 2 pulses, because as previously said in the *initial research* section, ones and zeros are encoded as the pulse to gap ratio.

The maximum data pulses allowed is flexible but is bounded up to 128 pulses, that is 64 bits, which is the maximum data length ever received from a test subject. Moreover, a minimum is also imposed to reduce environmental noise. As the minimum data length ever received from a test subject is 32 bits, anything less than 64 pulses will be discarded.

The protocol parameters for each test subject are hardcoded in the code on an array following the C zeroed-initialization style "{Width, {Sync}, {Preamble}, {{Zero}, {One}}}".

```
static const Protocol PROTOCOLS[] = {
    {400, {1, 42}, {1, …(21)…, 1, 10}, {{1, 2}, {2, 1}}}, // Wireless motor. blinds
    {300, {1, 34}, {}, {{1, 3}, {3, 1}}},                 // Wireless socket
    {450, {1, 29}, {}, {{1, 3}, {3, 1}}},                 // Open-close alarm
}
```

Finally, the data actually received or transmitted is encoded in the form of a "Packet". The implementations is as follows:

```
class Packet
{
public:
    uint8_t Protocol;
    char Data[MAX_DATA_PULSES/MAX_DATA_BIT_PULSES + 1];
};
```

Protocol describes the $index + 1$ (id) of the used protocol in the "PROTOCOLS" array above.

Data is defined as bits in a sequence of characters, also known as a string. It is a string and not an integer because the algorithm has been implemented with flexibility in mind. That is, if the data transmitted is larger than an integer (32-bit) or a long integer (64-bit) it can still be received and transmitted flawlessly. The string is obviously bounded, as this is an embedded system with resource limitations, but it is flexible, the maximum allowed characters (65) is the maximum allowed data pulses (128) divided by the maximum allowed bit pulses (2) plus one more character that corresponds with the "\0" null-terminator.

Device receiver

The receiving algorithm is implemented as an interrupt service routine that continuously fills a bounded circular buffer with the widths of the received pulses. The circular buffer size (156) is the sum of the maximum allowed sync pulses (4) plus the maximum allowed preamble pulses (24) plus the maximum allowed data pulses (128).

When a pulse is longer than the smallest known sync pulse, within a certain tolerance (25%), it is probably the start of a signal. When the sync pulse is matched again it probably is the end of the signal and the start of a new one, so it tries to decode the received data by matching every known protocol model against the full received pulse sequence, again with a certain tolerance (25% different).

If every phase matches it finally publishes the resulting packet into a queue for later use. The queue length is 25 packets.

The following *Figure 146* illustrates an abstract implementation of such algorithm divided in two flowcharts. The second flowchart is the implementation of the "Decode data" function of the first flowchart.

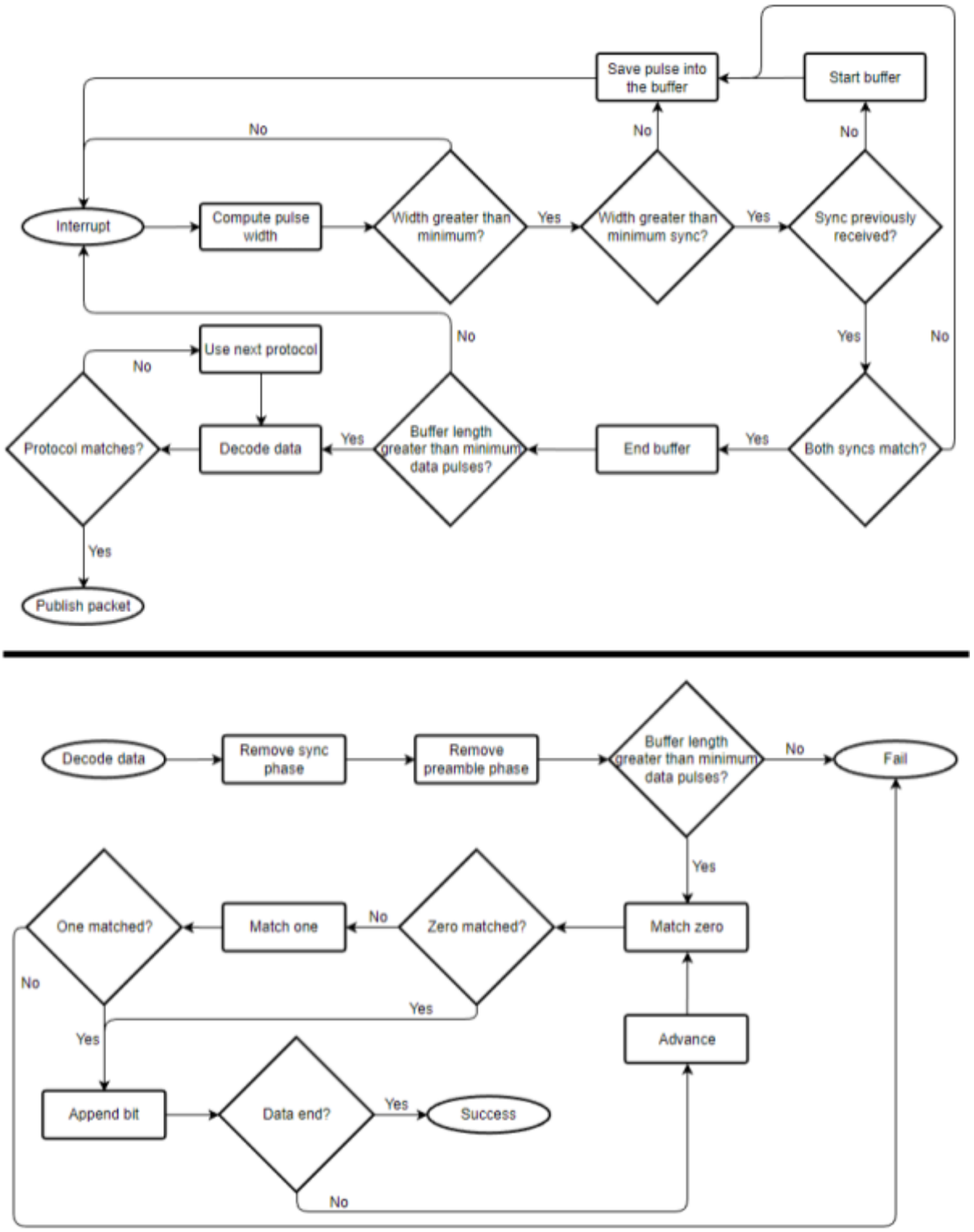


Figure 146: Device receiver algorithm. *Source: Compiled by the author*.

Device transmitter

The transmitting algorithm is implemented as a task that constantly consumes a queue where packets are published to be sent. The queue length is 25 packets.

It builds the correct signal using the packet's known protocol implementation and the packet's data bits by toggling the transmission in every pulse of every phase. The transmitter task does all of this within a critical section, in order not to cut the transmission suddenly and ensure signal integrity.

The following *Figure 147* illustrates an abstract implementation of such algorithm divided in two flowcharts. The second flowchart is the implementation of the "Send pulses" function of the first flowchart.

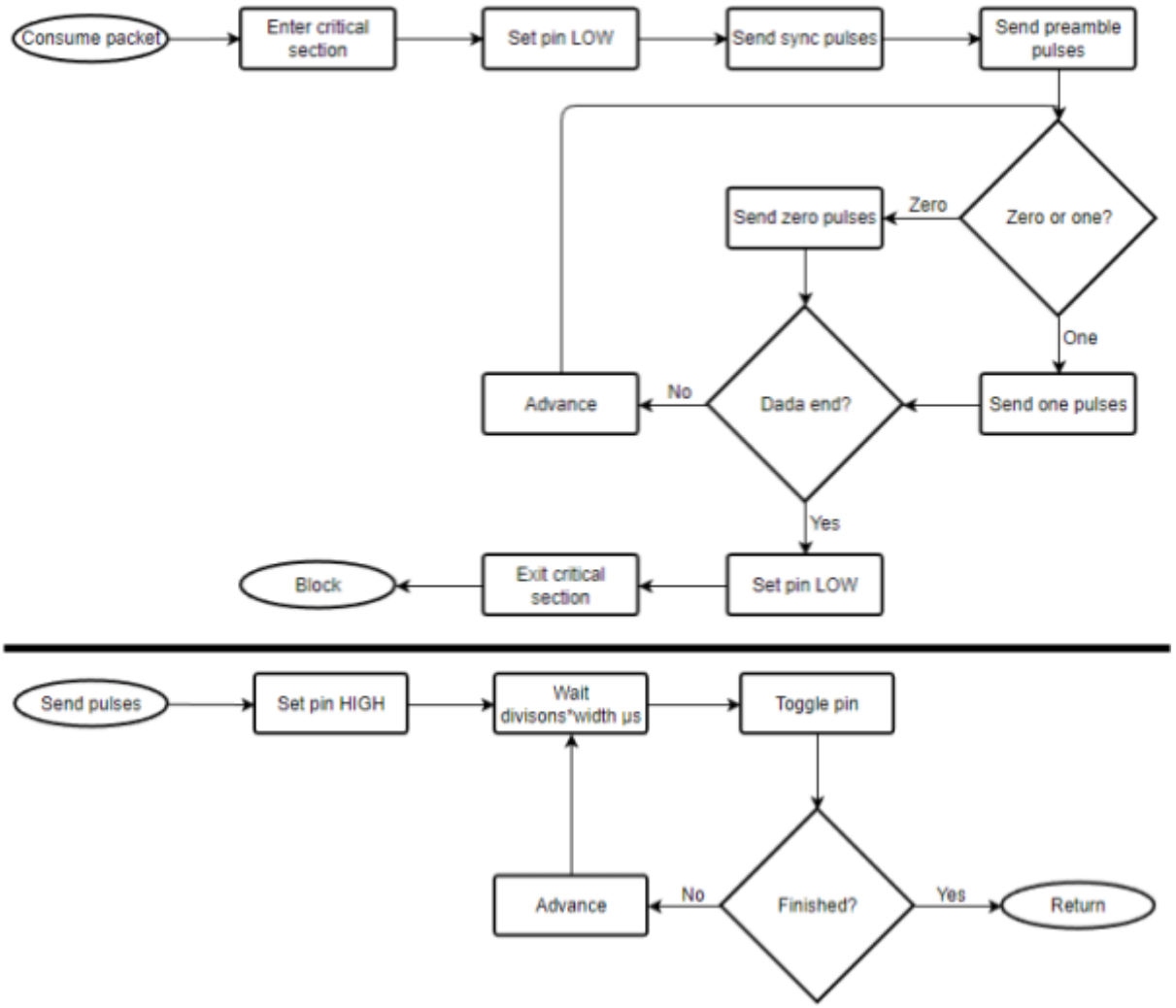


Figure 147: Device transmitter algorithm. *Source: Compiled by the author*.

Wi-Fi provisioner

The Wi-Fi provisioner is implemented as a task that initially tries to connect to the saved target Wi-Fi network or create directly an own softAP network. After that, every five minutes the provisioner tries to connect or reconnect to the saved target Wi-Fi network or fallback to the softAP network. If the device is unable to connect after ten retries, the provisioner fallbacks to the softAP network.

The following *Figure 148* illustrates an abstract implementation of the provisioning algorithm using a flowchart. The implementations for the functions "Start softap", "Stop softap", "Start station", "Stop station" have been omitted as they include interfacing with the underlying Wi-Fi driver APIs and is very long. However, if more context is wanted the *GitHub repository* [67] can be checked as it is fully commented.

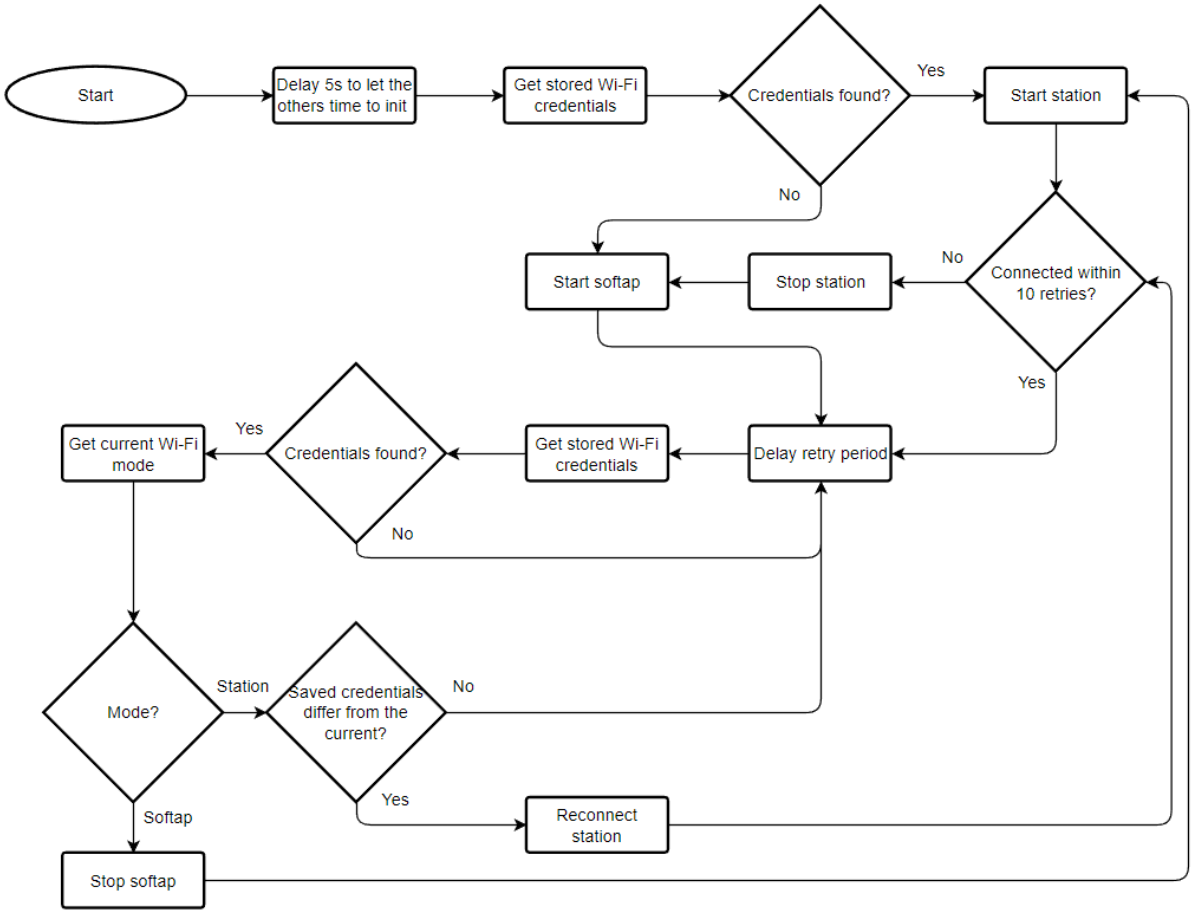


Figure 148: Wi-Fi provisioner algorithm. *Source: Compiled by the author*.

Status controller

The status RGB LED controller is implemented as a task that constantly consumes a queue where requests to change the RGB color are published. The queue length is 25 requests. Between each request there is at least a minimum wait of 125 milliseconds in order to ensure every color change is actually visible. The LED starts switched off, and as previously explained in the *taken approach* section, statuses are executed concurrently, that means, a request can hijack the current status and then switch again to the previous status if that status was configured to be "looping". The show time of the statuses can be either bounded or unbounded and can also blink by toggling the LED.

The following *Figure 149* illustrates an abstract implementation of such algorithm using a flowchart.

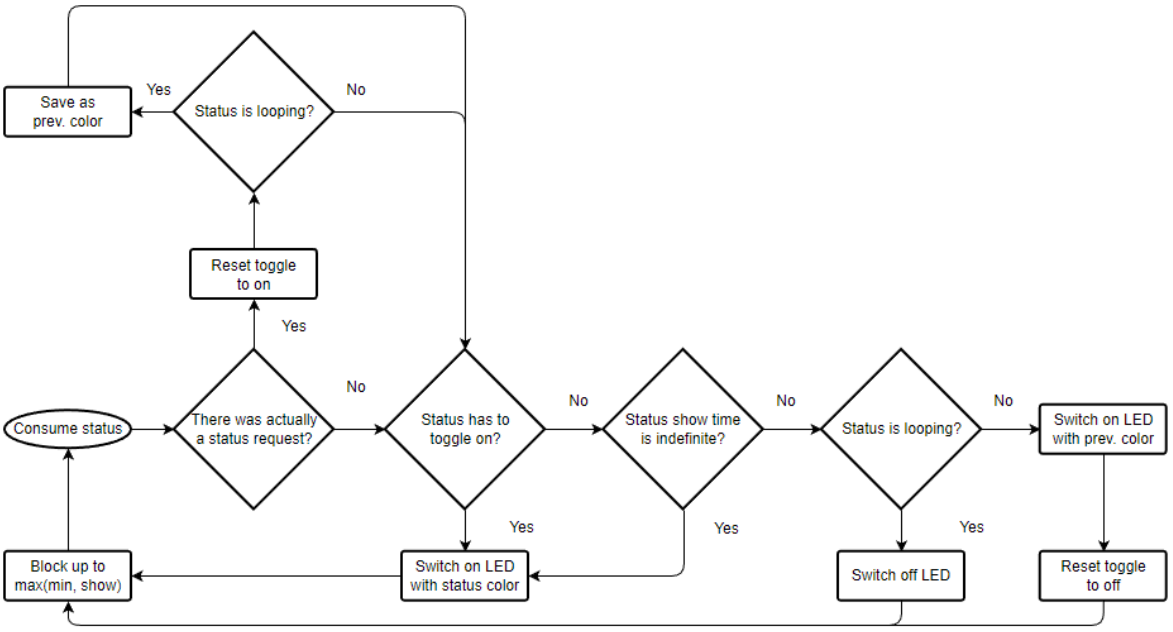


Figure 149: Status controller algorithm. *Source: Compiled by the author*.

Time synchronizer

The time synchronizer is implemented as a task that every five minutes takes the time from the RTC and adjusts the system time smoothly via the "adjtime" libc function, and that every hour requests an external server "pool.ntp.org" through NTP the current time and adjusts the RTC time immediately. A flowchart is not included as it wont contribute that much to the previous explanation.

Trigger scheduler

The trigger scheduler is implemented as a task that every minute tries to match every saved trigger schedule cron expression against the current time. If it matches, the respective actuator is triggered by sending a packet with the respective command to the transmitter queue. A flowchart is not included as it wont contribute that much to the previous explanation.

Frontend serving

The frontend endpoint matches the root path ("/") and catches all the other not matched paths ("/*").

If the requested path is the root one, the embedded gzipped frontend is served from the FAT filesystem of the static partition with the following HTTP cache control headers to prevent browsers from requesting the application each time a user accesses it so clients load the frontend faster.

a) Max Age: the response remains fresh until 604800 seconds (7 days) after the response is generated.

b) Stale While Revalidate: the cache can reuse a stale response while it revalidates it to a cache up to 86400 seconds (1 day).

c) Stale If Error: the cache can reuse a stale response when an upstream server generates an error when revalidating it up to 86400 seconds (1 day).

If the requested path is not found, the server redirects the request to the onboarding page if there are no users (ignoring the System default), otherwise to the frontend hash-based navigation " /#/:path". If the provisioner mode is softAP HTTP temporary redirects are used (302), otherwise permanent redirects are used (301) to prevent browsers from requesting this path again and go directly to the already potentially cached frontend. If the requesting user agent header is from an iOS device the redirect is served with some content, required so that devices detect the captive portal if being used.

A flowchart is not included as it wont contribute that much to the previous explanation.

CRUD serving

The API endpoints match "/api/users/*", "/api/devices/*", "/api/triggers/*", "/api/roles/*" and "/api/system/*" paths.

The following *Figures 150, 151, 152, 153* illustrate abstract implementations of a create (POST), read (GET), update (PUT) and delete (DELETE) HTTP REST API requests of the trigger model to exemplify how all the other API endpoints are implemented, as there are more than 30 available. Notice that both authentication and role based access control is performed in every request as required by the objectives.

System endpoints allow to retrieve the device's connected network (.../wifi) and its time (.../time), manage system configuration options (.../info) and factory reset the system (.../reset) which erases all content of the database and restarts the device.

Figure 150: Create API endpoint algorithm. *Source: Compiled by the author*.

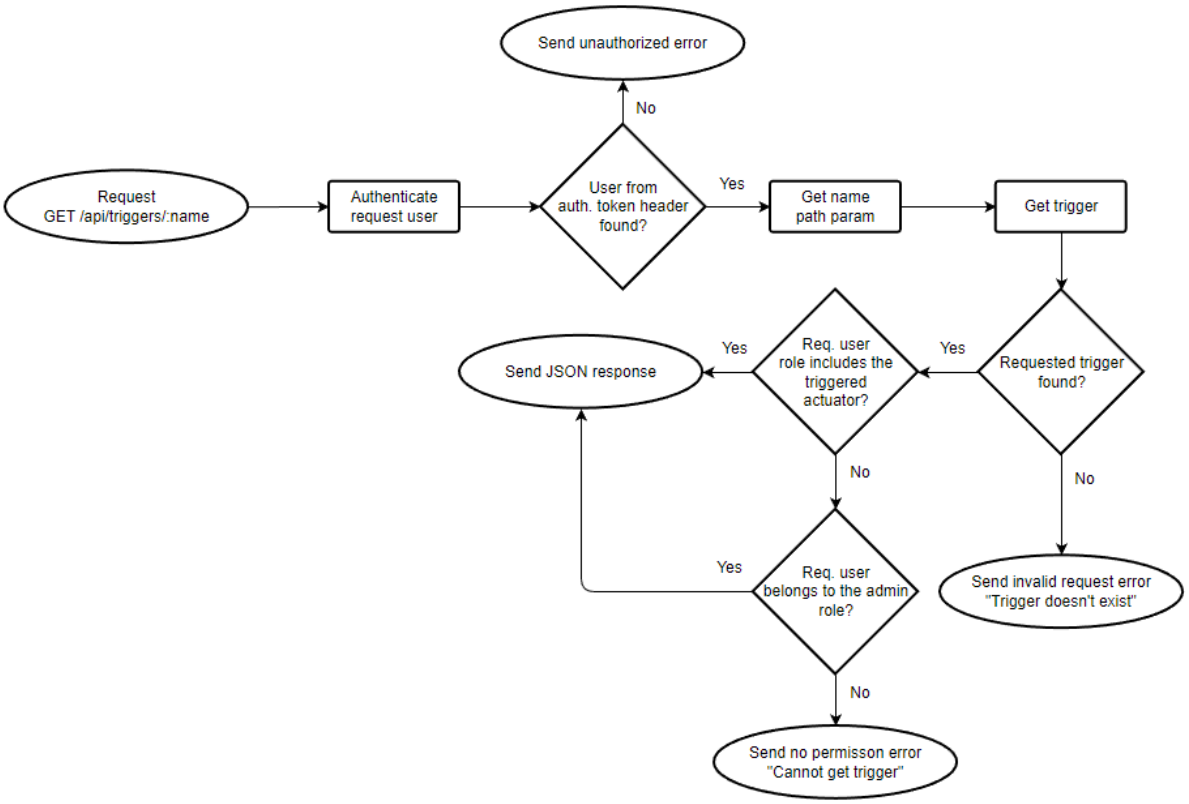


Figure 151: Read API endpoint algorithm. *Source: Compiled by the author*.

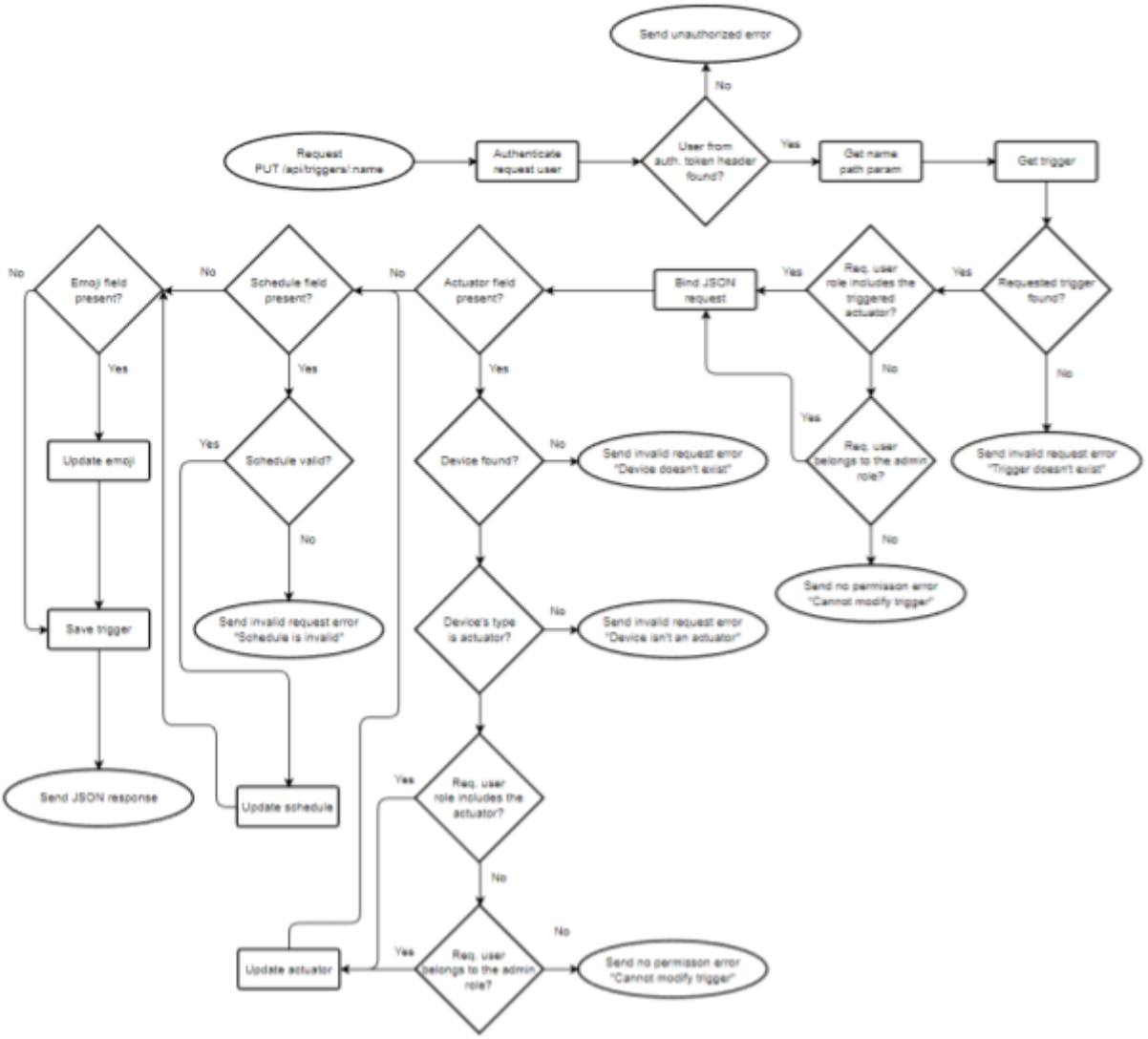


Figure 152: Update API endpoint algorithm. *Source: Compiled by the author*.

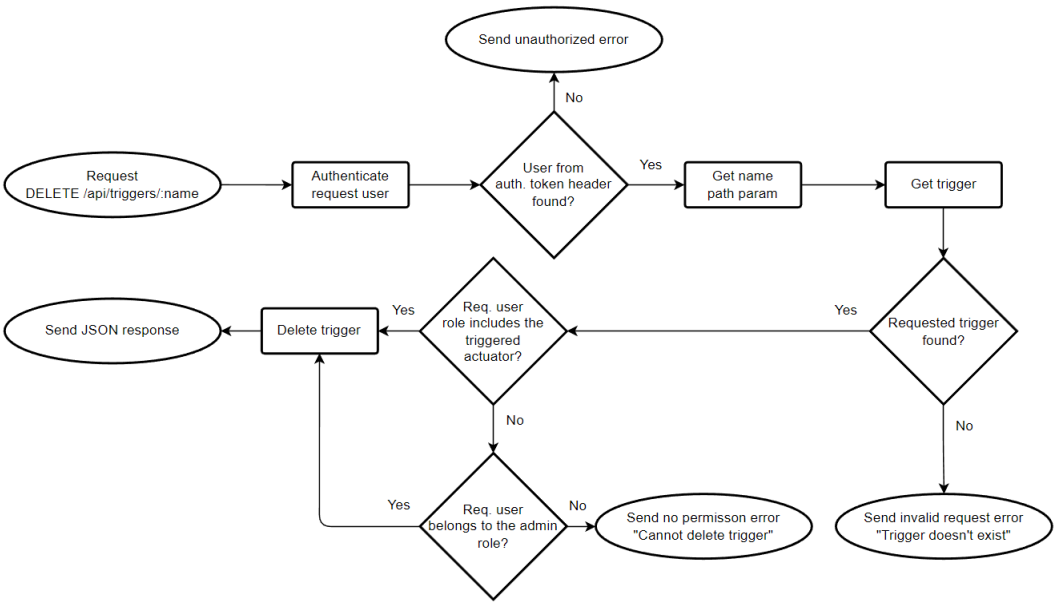


Figure 153: Delete API endpoint algorithm. *Source: Compiled by the author*.

14.2. Testing

Although no automatic unitary tests were written, as deadlines were tight, end to end tests of every endpoint were performed manually after finishing each one, using an extension from the development environment used VsCode, as seen in the following *Figure 154*.

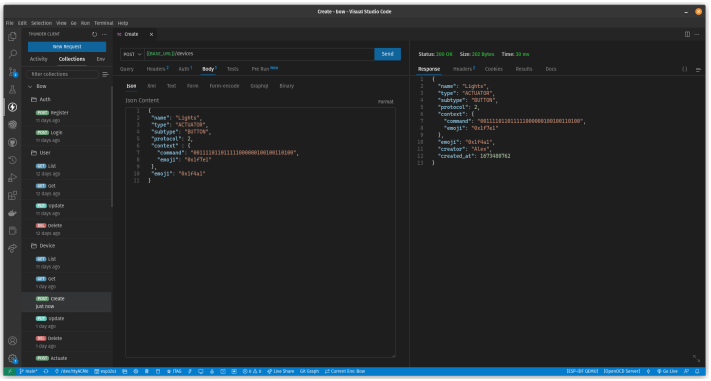


Figure 154: End to end testing of the backend firmware. *Source: Compiled by the author*.

These end to end tests are saved inside the “docs” directory of the *GitHub repository* [67] and can be actually reproduced.

Moreover, the compilation was obviously another testing checkpoint. Additionally a c/c++ linter was introduced early in the development to perform static analysis on the source code. It allows not only to follow a certain programming style but catches errors such as odd unused variables or functions, always true or false conditions, or even potential memory leaks as shown below in *Figure 155*.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
                ^
12/19 files checked 65% done
Checking components/update/update.cpp ...
13/19 files checked 65% done
Checking components/user/controller.cpp ...
components/user/user.hpp:44:5: style: The class 'Controller' does not declare a constructor although it has private member variables which likely require initialization. [noConstructor]
    class Controller
    ^

components/user/user.hpp:30:9: style: Class 'User' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
        User(cJSON *src);
        ^
14/19 files checked 68% done
Checking main/main.cpp ...
main/main.cpp:19:5: style: The class 'App' does not declare a constructor although it has private member variables which likely require initialization. [noConstructor]
    class App
    ^
```

Figure 155: Linting of the backend firmware. *Source: Compiled by the author*.

Finally the JTAG interface through the USB-C connection was used whenever a problem or bug was very hard to debug with log statements, or I wanted to make sure that a certain part of the code was doing what it was supposed to do, as seen in the following *Figure 156*.

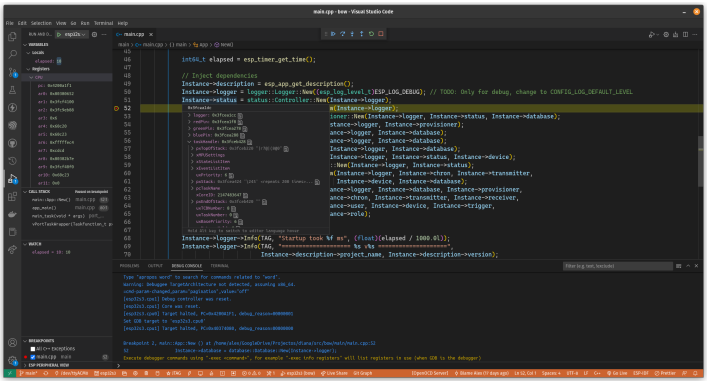


Figure 156: USB-C JTAG debugging of the backend firmware. *Source: Compiled by the author*.

14.3. Problems and solutions

As with any complex code done, or not, in limited time, there were errors and bugs at every stage of the development. Thats why, more than errors they were obstacles which eventually all were surpassed.

Without a doubt, the most annoying and common error was memory leaks or segmentation faults because of the manual memory management of c/c++. The big problem here is that the microcontroller crashes but does not generate any trace in order to be able to pinpoint the crashing statement, it must be manually debugged using log statements or spinning up the JTAG debugging interface.

In order to fix such errors dynamic memory requested via the “malloc” libc function must be liberated by the “free” libc function eventually or the memory will be consumed little by little until the point the microcontroller crashes as it cannot do a certain action.

Another interesting obstacle was to get the captive portal working on iOS as the problem could not be narrowed down to a single cause. Luckily, by testing and testing, the problem ended up being that the catch-all redirect scheme of a captive portal must contain some sort of content in order to trigger such on iOS.

Finally, another critical problem was the inability to encrypt the static partition where the frontend was embedded. That would cause the flash encryption requirement to be partially unachieved. The problem happened because the partition was initially using SPIFFS as the file system, by switching it to a FAT file system the whole flash memory could be encrypted.

14.4. Results

As said at the beginning of this section the embedded interface of the device could not be fully finished as there was no time left to do so. Hence the following *Figure 157* only shows debug console output to state the results. However, a full demonstration video controlling the system through the HTTP REST API is included later on in the *final results* section.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

D (28126) device: Rx | Data: 010010001101110000000011000011010 | Protocol: 3
D (28136) device: Received data from Alarm
D (28286) device: Rx | Data: 010010001101110000000011000011010 | Protocol: 3
D (28286) device: Received data from Alarm
D (28436) device: Rx | Data: 010010001101110000000011000011010 | Protocol: 3
D (28436) device: Received data from Alarm
D (62206) trigger: Actuator Lights-On triggered by Wake
D (62356) device: Tx | Data: 001111011011111000000100100110100 | Protocol: 2
D (68416) device: Rx | Data: 001111011011111000001110100101010 | Protocol: 2
D (71206) device: Rx | Data: 001111011011111000001011100100000 | Protocol: 2
D (71306) device: Rx | Data: 001111011011111000001011100100000 | Protocol: 2
D (72026) device: Rx | Data: 001111011011111000001010100100010 | Protocol: 2
D (72126) device: Rx | Data: 001111011011111000001010100100010 | Protocol: 2
D (122216) trigger: Actuator Lights-Off triggered by Sleep
D (122376) device: Tx | Data: 001111011011111000001000100100100 | Protocol: 2
```

Figure 157: Complete backend firmware. *Source: Compiled by the author*.

15. Final Results

15.1. Demonstration

The final result of this Thesis project is indeed the designed and implemented 433MHz gateway. The requirements imposed and the set objectives have been fully achieved with the exception of the embedded web application interface as it could not be fully finished in time. However the device can still be used with all of its features through the HTTP API manually.

The video attached as a link in the bibliography **[69]** of this Thesis project is a demonstration of the resulting device. Goes from its packaging "unboxing" and the exploration of every element to the usage of every feature implemented via the HTTP API.

The following *Figure 158* depicts all designed elements (except the firmware obviously) side by side.

Figure 158: Complete Thesis project. *Source: Compiled by the author*.

Finally, the power consumption of the final device was measured by taking the average consumption within a thirty minute usage of the system. A USB multifunction tester was used to perform the analysis, as seen below in *Figure 159*. The resulting wattage of the device is 0.4W or in battery terms 40mAh.

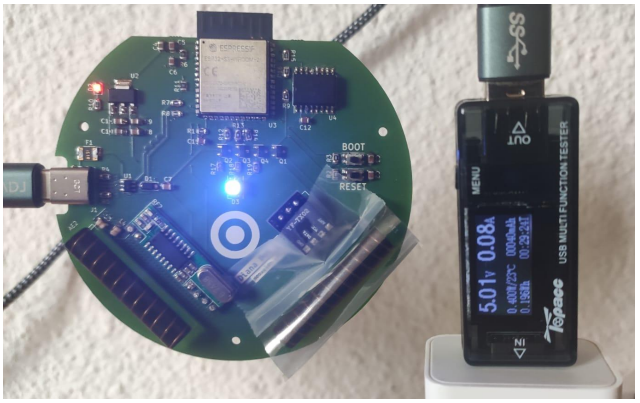


Figure 159: Power consumption of the device. *Source: Compiled by the author*.

15.2. Achievement of goals

Finally, at the end of the Thesis project, one must take a look back and check whether the initial requirements imposed and the set objectives have actually been achieved.

Schematic

All the objectives from the schematic aspect have been fully achieved. The most suitable hardware components were analyzed, selected and bought. The electrical schematic was designed and completed without any ERC errors.

PCB:

All the objectives from the PCB aspect have been fully achieved. The printed circuit board was designed and completed without any DRC errors. A 3D preview of the PCB without any missing hardware component was created and even renderized to produce images for this Thesis document. A PCB sample was manufactured without any electrical flaws, and it works flawlessly.

Casing

All the objectives from the casing aspect have been fully achieved. The casing was designed and a 3D preview of both the body and the lid was created and even renderized to produce images for this Thesis document. A case sample was manufactured without any measurement or mechanical errors.

Packaging

All the objectives from the packaging aspect have been fully achieved. The packaging was designed and a 3D preview of it was created and even renderized to produce images for this Thesis document. A package sample was manufactured without any measurement or styling errors.

Firmware

Almost all the objectives from the firmware aspect have been fully achieved. As there were a lot of objectives they will be referenced by their letters (check the *objectives* section). The objectives “a”, “b”, “c”, “d” (partially with the exception of the web app interface), “e”, “f”, “g”, “h”, “j”, “k”, “l”, “p” and “q” were achieved. “I” (OTA updates) was very complex, time consuming and zero rewarding. The firmware can be perfectly updated through USB-C via the JTAG interface. “M” (built-in web dashboard) is actually partially achieved as the device can indeed serve an embedded web application but its interface is not finished. “n”, “o” and “r” (TLS layers) were too resource intensive for the microcontroller and weren’t worth it.

16. Conclusions

16.1. Summary and personal opinion

I strongly believe that this project was a great success and it does solve with clearance all the core proposed objectives and requirements.

This Thesis project was not to just design or build a bare-simple hardware prototype, but rather taking a step above, contemplating additional aspects of a retail electronic device such as manufacturing the PCB, the casing and even the packaging.

That's why the final outcome is a very professional product prototype that could potentially be bought in a general consumer electronics shop.

The final user of the device can connect her/his 433MHz sensors and actuators to the gateway, receive data and transmit commands to them respectively, securely, reliably and through the Internet. Although not from the initially proposed web-based dashboard but from an HTTP API. Moreover, a trigger of a command based on a schedule can be created in order to perform an action on an actuator periodically.

The gateway allows to enhance the current RF devices that people has all around their houses and are important enough for them to rely and use them on a day to day basis, simplifying the communication and managing of these devices.

As a self-defined maker and a person that loves to deep dive into a problem that people actually have, understand it in its fullest, creating a solution that people will use and doing all of this while joining my two passions, software and hardware, I have not only enjoyed while doing the Thesis but it also step up my game in areas that I don't feel comfortable with, such as designing PCBs or CAD models, that in the professional world are must-to-have skills.

I felt completely capable and comfortable working on it every day, improving it little by little and seeing my effort come to fruition. I was able to plan and create a project that not only genuinely interested me but it was a concept that would help others.

This Bachelor's Thesis is by far the largest project I have ever designed and implemented by myself. Although there were moments and mistakes that definitely set me back, I learned from those failures, and, in the end, made the project better and achieving every core goal I set out for it

I am proud of the outcome of my ambitious final degree project and am motivated to continue pursuing my field, expanding my knowledge, and making a positive impact in the industry.

16.2. Future work

Actually, there is not much future work left from this Thesis aside from:

a) Fixing the flaws presented in every *problems and solutions* sections directly in the designs themselves. That includes adding pull-down resistors on the transistors, removing the offending boot capacitor and the grounded inductor, reversing the transmitting module footprint, designing a better RF isolation solution and adding some tolerance to the snap-fit closing mechanism.

b) Finishing the OTA updates feature and the built-in web application interface, as there was not enough time to complete them, to fully "end" the Thesis goals.

Moreover, although not that important as the device is wired, the power consumption can be reduced furtherly by, for example, switching off selectively different parts of the device when not used, such as Wi-Fi or RF communications, or even reselecting other less consuming components.

Finally, more features could be added, such as a battery and an automatic charging system for the real time module or even polishing every design to make it mass manufacturable if a real product and a company is wanted to be made from this Bachelor's Thesis final project.

17. Bibliography

[1] Jovanovic, B. (n.d.). *Internet of Things statistics for 2022 - Taking Things Apart*. Retrieved September 24, 2022, from https://dataprot.net/statistics/iot-statistics/

[2] Hasan, M. (n.d.). *State of IoT 2022*. Retrieved September 24, 2022, from https://iot-analytics.com/number-connected-iot-devices/

[3] Choi, K., Shin, C., Kim, T., Chung, H. J., & Suk, H. J. (2019). Awakening effects of blue-enriched morning light exposure on university students' physiological and subjective responses. *Scientific Reports 2019 9:1*, *9*(1), 1–8. https://doi.org/10.1038/s41598-018-36791-5

[4] Clark, J. (n.d.). *What is the Internet of Things, and how does it work?* Retrieved September 24, 2022, from https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/

[5] *LPD433 - Wikipedia*. (n.d.). Retrieved September 24, 2022, from https://en.wikipedia.org/wiki/LPD433

[6] *Amplitude modulation - Wikipedia*. (n.d.). Retrieved September 24, 2022, from https://en.wikipedia.org/wiki/Amplitude_modulation

[7] *Amplitude-shift keying - Wikipedia*. (n.d.). Retrieved September 24, 2022, from https://en.wikipedia.org/wiki/Amplitude-shift_keying

[8] *What Is a Network Gateway? - Cisco*. (n.d.). Retrieved September 24, 2022, from https://www.cisco.com/c/en/us/products/routers/what-is-a-network-gateway.html

[9] *Hypertext Transfer Protocol - Wikipedia*. (n.d.). Retrieved September 24, 2022, from https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[10] *What is a REST API? | IBM*. (n.d.). Retrieved September 24, 2022, from https://www.ibm.com/cloud/learn/rest-apis

[11] *Google Trends*. (n.d.). Retrieved September 25, 2022, from https://trends.google.com/trends

[12] Cyrus, C. (n.d.). *IoT Cyberattacks Escalate in 2021, According to Kaspersky*. Retrieved September 25, 2022, from https://www.iotworldtoday.com/2021/09/17/iot-cyberattacks-escalate-in-2021-according-to-kaspersky/

[13] *Salario Medio en España (Sueldos 2022) | Jobted.es*. (n.d.). Retrieved October 7, 2022, from https://www.jobted.es/salario

[14] *DigiKey Electronics Spain - Electronic Components Distributor*. (n.d.). Retrieved October 7, 2022, from https://www.digikey.es/en

[15] *AISLER - Powerful Prototyping made in Germany*. (n.d.). Retrieved December 7, 2022, from https://aisler.net/?lang=en

[16] *Create online custom printed packaging, dielines and 3Ds*. (n.d.). Retrieved December 7, 2022, from https://www.pack.ly/en

[17] *Amazon.es: compra online de electrónica, libros, deporte, hogar, moda y mucho más.* (n.d.). Retrieved October 9, 2022, from https://www.amazon.es/

[18] *Precio de la tarifa de luz por horas*. (n.d.). Retrieved October 10, 2022, from https://tarifaluzhora.es/?tarifa=pcb&fecha=10%2F10%2F2022

[19] *Comparador de Ofertas de Internet | Octubre 2022*. (n.d.). Retrieved October 10, 2022, from https://www.kelisto.es/internet

[20] *Inicio - MADE Makerspace Barcelona*. (n.d.). Retrieved October 10, 2022, from https://made-bcn.org/es/

[21] *The Most Affordable Coworking Spaces in Barcelona*. (n.d.). Retrieved October 10, 2022, from https://itnig.net/blog/coworkings-barcelona/

[22] *Qüestionari Genèric Estudiants d'Enginyeria*. (n.d.). Retrieved October 10, 2022, from https://docs.google.com/forms/d/e/1FAIpQLSfVgBxcxZfh7pB_OVRUNGQmRpFDFlhAskukNcpQBowLRF4-sA/viewform

[23] *Transforming our world: the 2030 Agenda for Sustainable Development | Department of Economic and Social Affairs*. (n.d.). Retrieved October 10, 2022, from https://sdgs.un.org/2030agenda

[24] *General Data Protection Regulation - EUR-Lex - 32016R0679*. (n.d.). Retrieved December 8, 2022, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679&qid=1670531008195

[25] *Cyber Resilience Act | Shaping Europe's digital future*. (n.d.). Retrieved December 8, 2022, from https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act

[26] *Restriction of Hazardous Substances - EUR-Lex - 52002AG0019*. (n.d.). Retrieved December 8, 2022, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52002AG0019&qid=1670532983983

[27] *Electromagnetic Compatibility - EUR-Lex - 32014L0030*. (n.d.). Retrieved December 8, 2022, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32014L0030&qid=1670533934061

[28] *Radio Equipment Directive - EUR-Lex - 32014L0053*. (n.d.). Retrieved December 8, 2022, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32014L0053&qid=1670534938043

[29] *CE marking – obtaining the certificate, EU requirements - Your Europe*. (n.d.). Retrieved December 8, 2022, from https://europa.eu/youreurope/business/product-requirements/labels-markings/ce-marking/index_en.htm

[30] *Waste Electrical and Electronic Equipment - EUR-Lex - 32012L0019*. (n.d.). Retrieved December 8, 2022, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32012L0019&qid=1670536226078

[31] *General Product Safety Directive - EUR-Lex - 32001L0095*. (n.d.). Retrieved December 8, 2022, from https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex%3A32001L0095

[32] *Common Charger Directive for electronic devices*. (n.d.). Retrieved December 8, 2022, from https://ec.europa.eu/commission/presscorner/detail/en/IP_21_4613

[33] Csete, A. (n.d.). *gqrx-sdr/gqrx: Software defined radio receiver powered by GNU Radio and Qt*. Retrieved September 23, 2022, from https://github.com/gqrx-sdr/gqrx

[34] Pohl, J. (n.d.). *jopohl/urh: Universal Radio Hacker: Investigate Wireless Protocols Like A Boss*. Retrieved September 23, 2022, from https://github.com/jopohl/urh

[35] *About RTL-SDR*. (n.d.). Retrieved September 23, 2022, from https://www.rtl-sdr.com/about-rtl-sdr/

[36] *KiCad EDA - Schematic Capture & PCB Design Software*. (n.d.). Retrieved October 1, 2022, from https://www.kicad.org/

[37] *Fusion 360 | 3D CAD, CAM, CAE, & PCB Cloud-Based Software | Autodesk.* (n.d.). Retrieved October 1, 2022, from https://www.autodesk.eu/products/fusion-360/overview

[38] *Figma: the collaborative interface design tool.* (n.d.). Retrieved October 1, 2022, from https://www.figma.com/

[39] *ESP-IDF Programming Guide - ESP32-S3 - — ESP-IDF Programming Guide release-v5.0 documentation.* (n.d.). Retrieved October 1, 2022, from https://docs.espressif.com/projects/esp-idf/en/release-v5.0/esp32s3/index.html

[40] *Visual Studio Code - Code Editing. Redefined*. (n.d.). Retrieved October 1, 2022, from https://code.visualstudio.com/

[41] *Arduino - Home.* (n.d.). Retrieved October 1, 2022, from https://www.arduino.cc/

[42] *FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions.* (n.d.). Retrieved October 1, 2022, from https://www.freertos.org/

[43] *lwIP: Overview.* (n.d.). Retrieved October 1, 2022, from https://www.nongnu.org/lwip/2_1_x/index.html

[44] Michaely, M. (n.d.). *The Ultimate Guide to Cron: Read and write cron expressions like a pro.* Retrieved October 1, 2022, from https://crontogo.com/blog/the-complete-guide-to-cron/

[45] *Svelte • Cybernetically enhanced web apps.* (n.d.). Retrieved October 1, 2022, from https://svelte.dev/

[46] *Farnell Spain - Electronic Components Distributor.* (n.d.). Retrieved November 7, 2022, from https://es.farnell.com/en-ES

[47] *ESP32-S3-WROOM2 Datasheet*. (n.d.). Retrieved November 7, 2022, from https://www.espressif.com/documentation/esp32-s3-wroom-2_datasheet_en.pdf

[48] *QAM-RX10-433 Datasheet.* (n.d.). Retrieved November 7, 2022, from https://media.digikey.com/pdf/Data%20Sheets/RF%20Solutions%20PDFs/QAM-RX10.pdf

[49] *QAM-TX2-433 Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.rfsolutions.co.uk/downloads/6377a8c5b35ac672DSQAM-TX2-5.pdf

[50] *W3127 Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/2614647.pdf

[51] *DS3231SN Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.mouser.es/datasheet/2/256/DS3231-1513891.pdf

[52] *CLX6F-FKC Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/2573293.pdf

[53] *BC847B Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/2861456.pdf

[54] *USB4085 Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/3621541.pdf

[55] *LDL1117S33R Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/2259188.pdf

[56] *MP008293 Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/3497876.pdf

[57] *MF-MSMF075/24 Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/2264697.pdf

[58] *1N5819HW-7-F Datasheet.* (n.d.). Retrieved November 7, 2022, from https://4donline.ihs.com/images/VipMasterIC/IC/DIOD/DIOD-S-A0011437026/DIOD-S-A0011683586-1.pdf?hkey=6D3A4C79FDBF58556ACFDE234799DDF0

[59] *USBLC6-2SC6 Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/1690354.pdf

[60] *KMR421G-LFS Datasheet.* (n.d.). Retrieved November 7, 2022, from https://www.farnell.com/datasheets/2969021.pdf

[61] *Multicomp Pro | The Engineer's Choice.* (n.d.). Retrieved November 7, 2022, from https://multicomp-pro.com/

[62] *Multilayer Ceramic Chip Capacitors | TDK Electronics.* (n.d.). Retrieved November 7, 2022, from https://product.tdk.com/en/products/capacitor/ceramic/mlcc/index.html

[63] *SMT inductors | TDK Electronics.* (n.d.). Retrieved November 7, 2022, from https://www.tdk-electronics.tdk.com/en/529794/products/product-catalog/inductors-coils/smt-inductors

[64] *ESP32-S3-DevKitC-1 v1.1 — ESP-IDF Programming Guide release-v5.0 documentation.* (n.d.). Retrieved November 29, 2022, from https://docs.espressif.com/projects/esp-idf/en/release-v5.0/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html

[65] *ESP32-S3 Hardware Design Guidelines.* (n.d.). Retrieved November 29, 2022, from https://www.espressif.com/sites/default/files/documentation/esp32-s3_hardware_design_guidelines_en.pdf

[66] *IC Basics: Comparison of Switching Standards.* (n.d.). Retrieved November 29, 2022, from https://www.ti.com/lit/sg/sdyu001ab/sdyu001ab.pdf

[67] Rodríguez Navas, A. (n.d.). *Neoxelox/bow: Diana Dot backend.* Retrieved January 10, 2023, from https://github.com/neoxelox/bow

[68] Rodríguez Navas, A. (n.d.). *Neoxelox/arrow: Diana Dot frontend.* Retrieved January 10, 2023, from https://github.com/neoxelox/arrow

[69] Rodríguez Navas, A. (n.d.). *Demonstration Video.* Retrieved January 10, 2023, from https://www.youtube.com/watch?v=QUFasvSojwE

This page is intentionally left blank.