Danmarks Tekniske Universitet

DTU

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Design and implementation of a monitoring system for power tools

Bachelor's Thesis

Author

Juan Marin Collado - **s221856**

Supervisor Professors

Associate professor Martin Nordal Petersen (*DTU* Electro)
Carlos Enrique Palau Salvador (Communications department *UPV*)

Abstract

The rising of *IoT* devices thanks to *Industry 4.0* has boosted the deployment of *LPWAN* networks and cheapened the price of devices. In the next years, is expected to experience a huge increment in connected devices as more and more different sectors have started to adopt it in their fields. One example is construction sites, where tools, equipment and assets are all over the place. There are already proprietary *IoT* solutions to track, manage and configure tools but rely on poor Bluetooth networks built by the app users. In addition, the series of tools that support them are new high-end products, leaving out plenty of models. Information about the internal components of tools and batteries is really limited. Most of the required information to build the prototype was acquired by measuring and testing. In this Bachelor's thesis, a monitoring add-on will successfully be designed and implemented from scratch. Using the *LoRaWAN* Helium network to send to the backend the location and monitor data. The location indoors will be obtained with *WiPS* and outdoors with a *GNSS* module. Downlink messages can be sent to set different configurations to the end device. The complete integration of the backend in Azure was not achieved but a live dashboard with the values was accomplished using *Power BI*.

Keywords: 3D modelling; 3D printing; Prototyping; LoRaWAN; Power tool; Geotracking; LILYGO TTGO LORA32; ESP32; Embedded system; WIFI scanning; PCB manufacturing;

Contents

Nomenclature **iv**

1 Introduction **2**

2 Theoretical overview **3**
2.1 Communication protocols . . . 3
2.1.1 NB-IoT . . . 4
2.1.2 LTE-M . . . 4
2.1.3 SigFox . . . 5
2.1.4 LoRa . . . 7
2.1.5 LoRaWAN . . . 9
2.1.6 Conclusion . . . 11
2.2 Geopositioning . . . 12
2.3 WiPS (WiFi Positioning System) . . . 13
2.3.1 WiGLE API . . . 13
2.3.2 Skyhook API . . . 14
2.4 3D design . . . 14
2.5 3D printing . . . 15
2.5.1 SLA vs FDM . . . 16
2.6 Printed Circuit Boards . . . 16

3 State of the art **18**

4 Requirements **24**

5 Add-on design **25**
5.1 Hardware selection . . . 25
5.1.1 Microcontroller . . . 25
5.1.2 GNSS module . . . 26
5.1.3 *DC-DC* regulator . . . 27
5.1.4 LiPo battery . . . 28
5.2 Printed Circuit Board . . . 29
5.2.1 PCB design . . . 29
5.3 Add-on behaviour . . . 32
5.4 LoRaWAN network provider . . . 35
5.5 Custom communication protocol . . . 35
5.6 3D design . . . 37
5.6.1 Revopoint Mini . . . 38
5.6.2 Zeiss GOM Inspect . . . 41
5.6.3 Autodesk Fusion 360 . . . 41
5.6.4 Modelling process summarised: . . . 43
5.7 Prototype production . . . 43
5.7.1 3D printers . . . 43

5.7.2 Material for printing . . . 45
5.7.3 PCB production . . . 45

6 Prototype evolution **48**
6.1 Battery connector . . . 48
6.1.1 Version 1 . . . 48
6.1.2 Version 2 . . . 48
6.1.3 Version 3 . . . 49
6.2 Power tool connector . . . 49
6.2.1 Version 1 . . . 49
6.2.2 Version 2 . . . 50
6.2.3 Version 3 . . . 50
6.3 Metal connectors . . . 51
6.3.1 Male . . . 53
6.3.2 Female . . . 53
6.4 PCB evolution . . . 54
6.5 Add-on prototype cost . . . 54

7 Microcontroller code **57**

8 Backend **58**

9 Testing **60**
9.1 Dimensions and weight . . . 60
9.2 Power tool consumption . . . 60
9.3 Add-on power consumption . . . 60
9.4 Heat dissipation . . . 63
9.5 GNSS module real measurements . . . 63
9.6 Skyhook performance test . . . 64

10 Conclusion **65**

11 Future Work **66**

Appendices **67**

A LILYGO TTGO LORA V1.3 schematic **67**

B WiFi scan at the DTU library **72**

C Java Script Deparser of the payload **72**

D JSON deparsed message **74**

E Query code **75**

F Microcontroller Arduino code **76**

List of Figures **I**

List of Tables **III**

References **IV**

Nomenclature

AES Advanced Encryption Standard

AP Access Point

API Application Programmed Interface

BLE Bluetooth Low Energy

BT Bluetooth

DC Direct Current

DC(Helium) Data Credits

DSL Digital Subscriber Line

DTU Danmarks Tekniske Universitet

FDM Fused Deposition Modeling

GNSS Global Navigation Satellite System

GPS Global Positioning System

GSM Global System for Mobile communications

IoT Internet of Things

ISM Industrial Scientific and Medical

LCD Liquid Crystal Display

LoRa Long Range

LoRaWAN Long Range Wide Area Network

LOS Line Of Sight

LTE Long Term Evolution

M2M Machine to Machine

MAC Media Access Control

NB-IoT Narrow Band Internert of Things

PC Personal Computer

PLA Polylactic acid

RFTDMA Random Frequency and Time-Division Multiple Access

RSSI Received Signal Strength Indicator

SIM Subscriber Identification Module

SLA Stereolithography

SSID Service Set Identifier

TTN The Things Network

UI User Interface

UNB Ultra Narrow Band

UPV Universitat Politècnica de València

UV Ultra Violet

WiFi Wireless Fidelity

WiPS Wi-Fi positioning system

1 Introduction

For leasing companies or businesses that own plenty of power tools, having them tracked at all times can be very useful and a money saver. With the incorporation of a monitoring system add-on this can be achieved, giving companies more control over their goods and many more possibilities. The proposed monitoring system consists of an easily installed, long-lasting piece in between the battery and the power tool. Most trackers right now are based on Bluetooth, not very integrated with the target device and rely on very poor networks to locate them. This ends with an accessible, exposed solution that can be easily spotted and manipulated. Furthermore, they run on one-use button batteries that need to be replaced every few years.

The proposed device through a microcontroller can position the target both outdoors and indoors. Then send the information using a reliable *LPWAN* to a backend and later display it or forward it to whom may concern. It is meant to share the battery with the power tool making it more sustainable and not having to specifically replace its power source. It can incorporate all kinds of sensors, enabling countless possibilities such as data analysis. This combined with advanced algorithms can detect abuse of the tool, prevent unnecessary damage and create many applications.

After a positive feasibility study [1], in this Bachelor's thesis research will be carried out to identify the most suitable technologies for the implementation of the add-on. Once the theoretical concerns are settled, the hardware will be selected accordingly. Trying to achieve a low power consumption, compact size and functional prototype, while documenting all the steps and considerations taken during the development. The goal is to produce a reliable add-on that outperforms the existing ones and listens to the needs of the users.

2 Theoretical overview

In order to execute this project successfully, it is important to consider certain theoretical aspects to determine the most suitable and up-to-date technologies to be used. These considerations include the evaluation of LPWAN technologies to efficiently communicate the add-on with the network, different geolocation methods to obtain a precise location, as well as 3D design, 3D printing and PCB design to join all the components. This way, a comprehensive theoretical framework is established to provide a solid foundation for the project and its implementation.

2.1 Communication protocols

For our use case, a Low-Power Wide Area Network (*LPWAN*) technology with long-range capabilities is required. For this reason, widely known technologies such as *WiFi*, *LTE*, *BT* or *BLE* are discarded for communication purposes due to their high power consumption and or low range 1. It also has to allow mobility of the device as the tool will not be in a fixed spot.

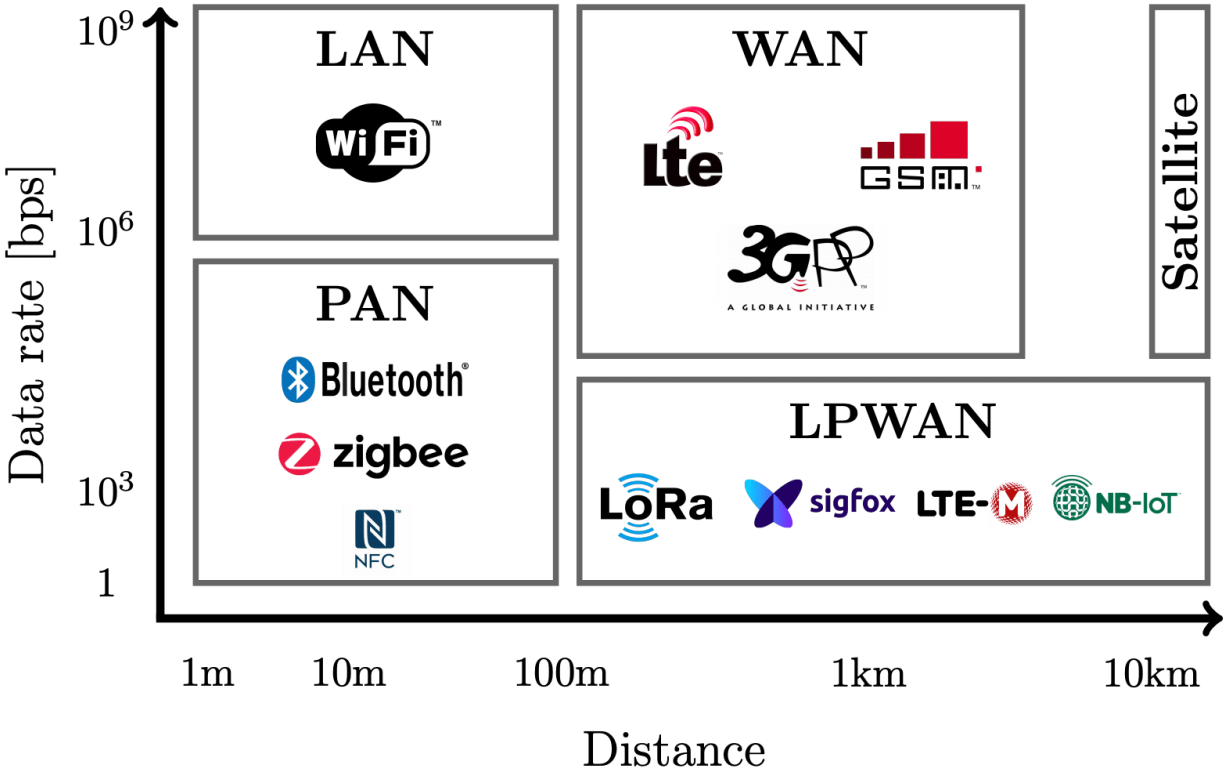


Figure 1: IoT Wireless Network Communication Protocols [2]

In recent years, with the large development and deployment of IoT networks, new long-range low-power telecommunication technologies have emerged. Providing us with a handful

of alternatives to choose from. Some would be: *SigFox*, *LoRaWAN*, *NB-IoT*, *LTE-M* ....

We can split them into two main groups: cellular and non-cellular. Cellular technologies (*NB-IoT*, *LTE-M*) use a licensed part of the frequency spectrum, and because the provider has paid to use those frequencies, they do not have any limitation in use. On the other hand, the non-cellular technologies (*SigFox*, *LoRaWAN*) make use of the open *ISM* band, with the drawback of duty-cycle restrictions as the spectrum resources are shared with many users.

2.1.1 NB-IoT

NB-IoT (Narrowband Internet of Things) is a technology developed by *3GPP*, particularly designed for wireless communication between IoT devices. It is under the Low-Power Wide Area Network (*LPWAN*) standard. *NB-IoT* is designed to operate in the licensed spectrum along with existing *GSM* mobile carrier networks and *LTE* technologies. It could be in an unused "guard band" or independently achieving up to 127 kbit/s in downlink and 159 kbit/s in uplink. Using the licensed spectrum ensures interference-free communication. Furthermore, as it uses the already existent cellular network, it provides a good range of connectivity without having to deploy a dedicated IoT network. This protocol will also be supported by *5G* towers. Allowing network operators to further monetize their investments, and increasing deployment scalability. NB-IoT security aspects are very alike to *4G* security. Including all encryption and *SIM*-based authentication features. In order to join the network, a *SIM* card and an operational plan are needed. The latency range is $1.6 \sim 10$ s.

NB-IoT networks are meant to support huge amounts of devices to fulfil the expected amount of *IoT* devices that will be deployed shortly, being able to connect billions of nodes. This is mainly thanks to the low production cost of devices. These have lower complexity than regular cellular modules, which brings down the bandwidth (low rates in *IoT*) and also helps to reduce its power consumption, making batteries last for years. NB-IoT mobility is reduced, therefore are is more prevalent in applications and use cases with stationary assets. Still an active protocol so future development may improve switching cells or even achieve handover.

2.1.2 LTE-M

LTE-M, is a Low-Power Wide Area Network (LPWAN) cellular technology optimized for Machine-to-Machine (*M2M*) and *IoT* applications. *LTE-M* is based on the *LTE* cellular standard, but modified to support lower power consumption and extended coverage, especially indoors and in rural areas. Thanks to this, it offers high data rates of up to *4Mbps* according to the Release 14 [3], meaning it can support Voice over *LTE* (*VoLTE*) and even low-resolution video. Very useful for surveillance purposes. *LTE-M* provides very low latencies of just tens of ms. In contrast with *NB-IoT*, *LTE-M* cannot be put into the guard band portion of the frequency band for now. It will continue evolving as part of the 5G specification, so further improvements are to come. Mobility is granted with handover allowing uninterrupted communication while changing base stations.

The security aspects are the same as for LTE. Use of *SIM*-based authentication, encryption and the private backhaul of the provider. Firmware updates can be sent Over The Air (OTA) so IoT solutions can be easily kept up to date.

2.1.3 SigFox

SigFox [4] is a proprietary ultra-narrow band (*UNB*) solution that only uses 192 KHz of the ISM band. In the case of Europe, from 868 to 868.2 MHz as seen in Figure 2.

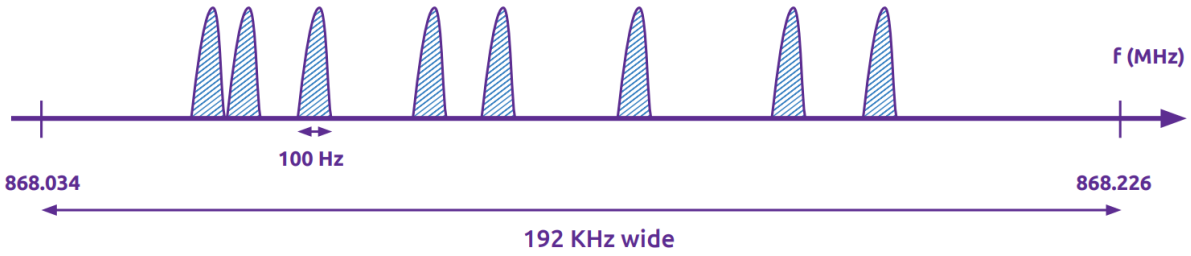


Figure 2: Sigfox technology based on Ultra-Narrow Band [5]

It is a protocol of small messages, with sizes going from 0 to 12 bytes of payload. Each message is modulated in a 100 *Hz* wide signal and sent with a data rate of 100 *bps*. For reference, a message with a 12-byte payload sent with a rate of 100 *bps* takes 2.08*s* over the air [5]. The regulation in Europe states that devices can occupy the ISM public band for 1% of the time. Meaning that each device can send up to 6 12-byte messages per hour, 140 messages per day.

For the uplink message, the device chooses a random frequency, sends the message and then 2 replicas are sent on different frequencies and times (see 3). This is called "time and frequency diversity" and it is used to ensure the quality of service and the reception of the message.

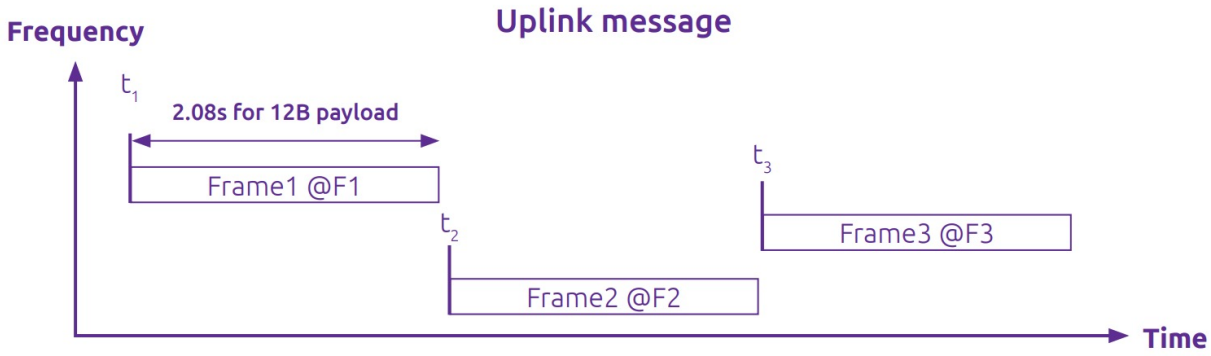


Figure 3: Frequency hopping on replicas[5]

Sigfox devices are not linked to a specific base station, so they don't communicate with it to keep their timing in sync. When a SigFox base station receives a message, it collaborates with other nearby stations to receive and understand the message. This is called cooperative reception. Each base station continuously monitors the entire 192 kHz frequency spectrum for Ultra Narrow Band (*UNB*) signals. When a signal is detected, it is demodulated and sent to the backend through a Digital Subscriber Line (*DSL*) connection. Once the message reaches the back end, it is processed, and the various versions of the message from different base stations are combined to eliminate duplicates and store only one copy of the message.

The core servers of the SigFox network monitor the network and manage the base stations globally. It stores the messages so the clients can access them through the web interface or integrate them in their *IT* system by using an *API*. Customers can also send downlink messages to individual devices.

The downlink messages are initialized by the device. It sends an uplink message with the "downlink request flag". Then the *SigFox* cloud does a callback to the customer *IT* server looking for any downlink payload for that specific device. If there is, the network transmits the downlink message that has a static payload size of 8 bytes. The frequency used is the same as the first uplink message plus a known delta. Base station regulations are less strict, having a 10% duty cycle which guarantees at least 4 downlink messages per device per day.

Altogether, the *UNB* modulation, plus the frequency and time diversity, and the overlapping network cells formed by the base stations, allows *SigFox* to have a massive device capacity. Besides, their chips are very energy efficient, consuming from 10 mA to 50 mA in transmission, and almost negligible when idle [5]. The max output power in Europe is 14 dB.

Another advantage of the *SigFox* technology is its long range, being able to cover large areas with a few base stations. The real range depends on the topography of the land, if there is line of sight (LOS) or not and the rate the message is sent (the lower the longer range). Besides, it has good indoor coverage as it uses sub *GHz* frequencies. Thanks to the

use of *UNB* signals and the "time and frequency diversity", SigFox has great resiliency to interferers in the public ISM band. For a message to be received by a base station, it needs to be at least 8 *dB* above the noise floor.

In the security aspect, authentication, integrity and anti-replay are guaranteed in the network. To achieve this, it uses Advanced Encryption Standard (*AES*) with a 128 bytes key and CTR mode to encrypt the transmitted messages. This encryption does not change the size of the payload, as the limit is 12 bytes and is decrypted by SigFox's cloud. In case the customer wants end-to-end confidentiality (from the device to the customer's cloud application), it must implement its own encryption mechanism. In this case, the SigFox network will not see the data in clear and will deliver it encrypted to the customer's application [6].

2.1.4 LoRa

LoRa (short for Long Range) is a proprietary radio communication technology developed by Cycleo and later acquired by Semtech. It transmits data using a chirp spread spectrum (CSS)-based proprietary modulation called Frequency Shift Chirp Modulation (*FSCM*) which does a great job in noisy means of transmission [7]. A chirp is a signal in which the frequency increases or decreases with time. Each chirp is a symbol. By nature, this *LoRa* chirp uses the frequency bands: 867-869MHz, 902-928 MHz and 430-510 MHz. Depending on the legislation where the devices are deployed, some bands or frequencies may not be used. The bandwidth implemented can either be 125 kHz, 250 kHz or 500 kHz.

Another variable is the spreading factor (*SP*) which controls the chirp rate. It can be any integer value between 7 and 12 (both included). Lower spreading values mean faster chirps thus less time on air and higher data rates. The downside is that the potential range also reduces with the dispersion factor because the SNR decreases and is more difficult for the receiver to decode the signal. In Figure 4 we can appreciate the correlation between these parameters.

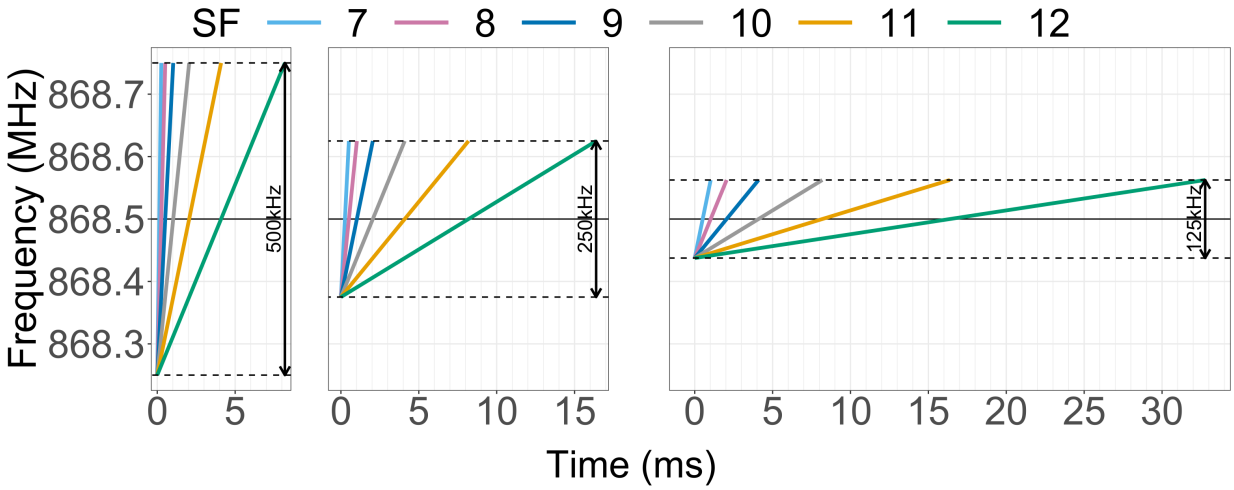


Figure 4: LoRa CHIRPs for SF $\in$ [7..12] and BW $\in [125, 250, 500] kHz$ on the $868.5 MHz$ channel [8]

Depending on the coding rate, the spreading factor and the bandwidth used, *LoRa* has different ranges and data transmission rates. Both, the sender and the receiver have to talk and hear respectively in the same *SP* to communicate effectively. *LoRa* reaches up to $5km$ in urban areas and $15km$ in suburban [9] and upload speeds from 980B/s to 21.9 KB/s. The max package size also varies and goes up to 242 Bytes [10].

In every communication, the sender first sends 8 preamble chirp-up pulses, next 2 sync symbols and then the payload. Example of a *LoRa* transmission in Figure 5.

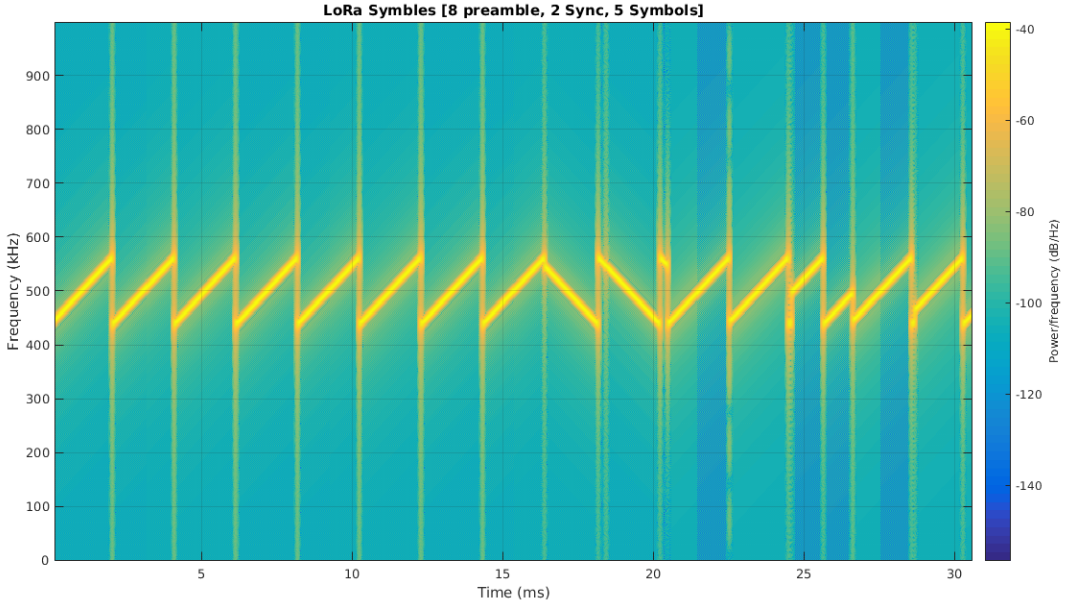


Figure 5: Example of a *LoRa* transmission [11]

LoRa by itself is only a way of wireless modulating the information. It only covers the first *OSI* layer. To enable more capabilities, we must use the *LoRaWAN* protocol that works on top of *LoRa* (see Figure 6). This way, we can register the device to an existent network, in which the gateways will receive the packages and forward them to the internet.

2.1.5 LoRaWAN

LoRaWAN defines the system communication and network architecture. It uses a star topology where the end-device sends the message and multiple gateways can receive it. Communication is bidirectional, although uplink messages from end nodes are more favoured as will be understood when talking about the three different LoRaWAN classes. Only supports the 912 MHz and 876 MHz bands from the *LoRa* physical layer.

As explained previously, there are duty-cycle limitations for end-devices on the 868 *MHz* European *ISM* band. It is established that they can only transmit for 1% of the time on each channel having to consider an appropriate spreading factor (the higher, the more time it takes to transmit and fewer messages can be sent).

LoRa modulation parameters must be tuned to achieve the minimum time on air but, at the same time, gateways need to receive enough SNR to decode the chirps. The protocol adapts and optimises the parameters making use of the Adaptive Data Rate (*ADR*) bite in the header.

Before an end-device can send or receive messages, it must be registered with a network. This procedure is known as activation. Once the device is activated, the end-n There are two activation methods available:

Over-The-Air-Activation (OTAA)

Is the most secure and recommended activation method for end devices. During OTAA, the device performs a join procedure with the network. First, sends a *Join Request* message with the fields:

- *AppEUI* 64-bit unique application identifier that uniquely identifies the entity able to process the request.
- *DevEU* 64-bit unique device identifier that identifies the end-device)
- *DevNonce* 2-byte value randomly generated by the end-device to keep track of their join requests. This way replay attacks are avoided.

Then, the network will process the request and if the request is accepted it will generate two session keys (*AppSKey* and *NwkSKey*) and a *Join Accept* reply message encrypted with the AES-128 bit *AppKey*. Finally, the end-node decrypts the *Join Accept* message and obtains both session keys (*AppSKey*, *NwkSKey*).

Activation By Personalization (ABP)

Is less secure than OTAA. With ABP, the device address and security keys ($AppSKey$,$NwkSKey$) are hardcoded into the device. This means that devices that use ABP cannot switch network providers without manually changing the keys in the device.

***LoRaWAN* device classes**

There are three different classes of devices depending on the features they have as we can see on the MAC layer in Figure 6.

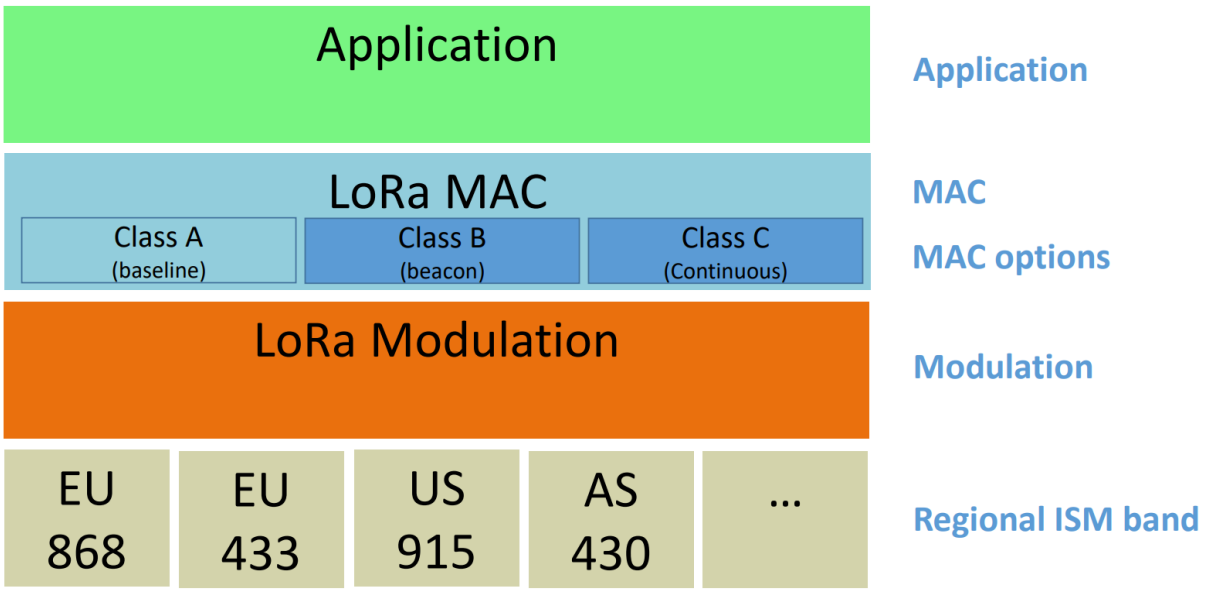


Figure 6: LoRaWAN Classes

Class A (Bi-directional end-devices): End-devices of *Class A* can handle bidirectional communication. Each uplink communication is followed by two short downlinks receive slots. The transmission slot schedule is based on the end-device communication needs plus a slight random variation introduced by the ALOHA-type protocol. This class is the most energy-efficient for end devices but at the same time, the slowest to communicate back to the end-device . Therefore is only meant for non-critic applications that require downlink communications shortly after an uplink transmission. Downlink communication at another time will have to wait until the next scheduled uplink. As it includes the basic features, it has to be supported by all *LoRaWAN-certified* devices.

Class B (Bi-directional end-devices with scheduled receive slots): End-devices of *Class B* allow more receive slots. In addition to *Class A* windows, this class allows extra receive windows at a scheduled time. As end devices do not tend to have internal clocks, the downlink windows are opened when they receive a time-synchronized beacon from the

gateway.

Class C (Bi-directional end-devices with maximal receive slots): End-devices of *Class C* have almost continuously open receive windows. They only close when the device is transmitting. This makes *Class C* end-devices the most energy-consuming of the three but also offer the lowest latency for server-to-end-device communication

Pseudo-random frequency is used to avoid collisions and to maximize the duty-cycle as limitations are per individual frequencies bands (see Table **??**).

Band		Duty cycles
K	(863 MHz - 865 MHz)	0.1%
L	(865 MHz - 868 MHz)	1%
M	(868 MHz - 868.6 MHz)	1%
N	(868.7 MHz - 869.2 MHz)	0.1%
P	(869.4 MHz - 869.65 MHz)	10%
Q	(869.7 MHz - 870 MHz)	1%

Table 1: European ISM duty cycle limitations per band [12].

2.1.6 Conclusion

Because the add-on is not a critical device and the aim is to stay as cheap and energy efficient as possible, cellular technologies are discarded. They require slightly more expensive hardware and monthly or yearly subscriptions. Also, the bandwidth provided exceeds by far the need for the project, and they drain more power. Furthermore, NB-IoT does not support mobility which is a must in this use case. The really strong point of these technologies is the network coverage they provide. But in global aspects, it is not the best option for this prototype implementation maybe for large deployments but that is out of scope now.

Taking into consideration the technical aspects, limitations and prices of both *LoRaWAN* and *SigFox*. Using *LoRaWAN* seems the best option. Only 140 12-byte packages a day is a low value for a monitoring system, especially ours which needs to send multiple WiFi information. Furthermore, *LoRaWAN* modules and subscriptions per device are cheaper than *SigFox* [13] or *NB-IoT / LTE-M*. Also, they are multiples network operators for *LoRaWAN* (i.e. Helium [14], TTN [15] or Cibicom [16]) not like *SigFox*'s single network. This enables the possibility of redundancy being able to try multiple network providers in case one of them does not have coverage in some areas and another does. LoRaWAN is widely implemented and has a huge community with forums and documentation.

2.2 Geopositioning

A global navigation satellite system (GNSS) is a constellation of satellites transmitting signal ranges used for positioning and location anywhere on the globe, whether on land, at sea or in the air. They allow the geographic coordinates and altitude of a given point to be determined as a result of receiving signals from constellations of artificial Earth satellites. It can be used for navigation, transportation, geodetics, localization and other related activities.

The constellations orbit between 20,000 to 37,000 kilometres above the Earth, in between the Medium Earth Orbit (MEO) and Geostationary Earth Orbit (GEO). These satellites broadcast signals that identify which satellite is transmitting, its time, orbit and status. Providing users high accurate position and time information, anywhere in the world, 24 hours a day and in all weather conditions. However, when the user/device is indoors attenuation and other phenomena affect the signal providing none or imprecise locations.

There are four main constellations in orbit each managed by a different country GPS (United States), GLONASS (Russia), Galileo (European Union) and BeiDou (China) as well as two regional systems QZSS (Japan) and IRNSS (India).

To determine the location of the satellites two types of data are required by the receiver: the *almanac* and the *ephemeris*. This data is continuously transmitted by the satellites and received by the module. The *almanac* contains information about the status of the satellites and approximate orbital information. The GPS receiver uses the almanac stored data to calculate which satellites are currently visible, this information can be valid for up to 90 days. Almanac is not accurate enough to let the GPS receiver get a fix.

The ephemeris data is very precise orbit information of each satellite. The receiver can use it to calculate the location of the satellite. Ephemeris information is valid for 4 hours. With 4 fixed satellites the module can get a 3-dimension position. If only 3 satellites are fixed, it can still retrieve a position but it will asume that the device is at sea level.

As we noted above, the GPS receiver needs 4 satellites to work out your position in 3-dimensions. If only 3 satellites are available, the GPS receiver can get an approximate position by making the assumption that you are at mean sea level.

The location obtained is specified by the latitude and longitude coordinates. A combination of these coordinates can identify any point on Earth. They can be displayed in degrees, minutes and seconds or as a float. Latitude measures how North or South from the equator a coordinate is. It ranges from -90 to 90 degrees and the 0Âş are in the equator. Longitude measures how East or West in the globe a coordinate is. It ranges from -180 to 180 degrees and the 0Âş are in the Greenwich meridian.

Different types of starts for GNSS modules

Cold start occurs when the GNSS module has not been used for a long time and/or has moved several hundred kilometres. The first fix will take several minutes. The receiver does not have any valuable initial almanac information about the position of satellites or time. The stored data will be incorrect and it will have to search again for them slowing down the process.

Warm start happens when the current almanac, initial position, and time are all valid. Ephemeris data is either invalid or only partially valid. The time to first fix can be a few minutes.

Hot start is when the receiver has been off for a small period of time and has valid alamac and ephemeris data. The expected time for the first fix will likely be a few seconds.

GNSS modules drain a lot of power (around $30mA$ [17]) and required an antenna to get the coordinates. They can be directly soldered to the module or external and connected via SMA or MHF. There are two kinds: passive and active antennas. Active is usually faster thanks to their low-noise amplifier that increases the gain, but it also drains 5-30mA [18] more power than passive ones, which do not drain any extra current but take longer to fix the signal.

2.3 WiPS (WiFi Positioning System)

It consists on retrieving a location in the globe by only knowing the surrounding *WiFi* access points. To achieve this, the device scans for the *WiFi* beacons that the AP broadcast. Then, it stores their *SSID*, *MAC* address, plus the Received Signal Strength Indicator (*RSSI*) and other known values. Finally, this information is parsed into a database where the location of lots of access points is stored, and with some calculations, an approximate location is obtained.

2.3.1 WiGLE API

WiGLE [19] is an open-source project where people worldwide can add hotspots and their locations. To do this, *WiGLEs* developed an app for smartphones, so anyone can start tracking all the *WiFi* signals available in their surroundings and send them to the main server alongside the position information.

They have also developed a JSON-based API to allow users to access WiGLE data for their own projects and get locations. Apart from WiFi hotspots, they also have the position of Bluetooth devices and cell towers to achieve more precise results.

2.3.2 Skyhook API

Skyhook [20] is a location solutions company recently acquired by Qualcomm. Is one of the most advanced providers of positioning software. They are now focused on the precise location of devices without the use of *GPS*. Very useful for our case and indoor tracking. To locate a device, you can use their API (key required) by sending information about surrounding: *WiFi APs*, *BLE beacons*, *GSM Cell Towers*, *UMTS/WCDMA Cell Towers*, *LTE Cell Towers*, *CDMA Cell Towers*, *NB-IoT Cell Tower*, and *NR Towers (5G, New Radio)*.

2.4 3D design

In the last decade, the use of 3D modelling tools in architecture and design industries has become a norm. From the first simple CAD programs all the way to the new cutting-edge software, a wide diversity of 3D modelling and rendering software for different purposes, workflows and budgets have emerged. Not only the number of available programs, but the list of products has likewise grown. Anyone who is an expert in engineering, structures, or design today can convey his ideas in a variety of fun ways. Virtual reality, panoramas, animations, photo-realistic images or video renders of interior and outdoor scenes. For beginners and freelancers, free-to-use programs such as Blender fit best. Other options are paid, some of them provide a cheaper or free version for students sometimes with limitations.

All combined allow the engineers to quickly design high-precision models, work as a team on the same project, thoroughly document each step in the assembly process, and simulate and test real work behaviour. Furthermore, it provides a realistic handful of ways of showing clients the expected product in all possible lightning and perspective thanks to powerful rendering engines.

For existing objects in which 3D design is not available, rather than measuring and trying to replicate the model, new methods are used. For example, photogrammetry [21] consists of extracting points from multiple pictures of an object. Is very important to get the images from different perspectives capturing all the details. Besides, the lightning has to be uniform in order to avoid artefacts or loss of detail. Large plain objects, dark surfaces or reflective materials are difficult to capture with this technique. After processing the images, the software retrieves a 3D design of the scanned area. Post-processing is required to extract only the desired object.

A similar technique is used but with laser images instead of RGB images. The scanning device projects a laser grid onto the object and the reflection is received by a sensor and an image is generated. It can be used in many situations such as building scanning [22] or it can be implemented in a portable way such as [23]. Laser images are more resistant to changes in natural light, and darker surfaces. Some implementations use both, laser and RGB images to take high-precision scans while conserving the RGB colours of the object [23].

Figure 7: Generative design of a piece with different levels of complexity [26]

The revolution of Artificial Intelligence (AI) also has a place in this field, enabling generative designs in which simple models are modified and optimized by AI. The user indicates the parameters that the model has to achieve and the software will try millions of options learning in each iteration until it finds an optimal solution. An example can be seen in Figure 7 where the evolution is shown from a base model to more complex designs. Researchers have achieved to generate 3D models out of sketches [24] and also from text prompts [25]. Many more tools are going to emerge and get improved in the near future. The 3D design will take advantage and adapt them to each field.

2.5 3D printing

3D printing is an industry that dates back to the 1980s. Methods of manufacturing plastics by hardening polymers with ultraviolet light were patented, and the first 3D printing method was devised, stereolithography. Other printing methods also appeared, from those using sintering to material extrusion.

3D printing is currently a developing industry with a cross-cutting technology, as it can be adapted to various fields such as medicine, fashion, architecture, design or engineering. It is very versatile, and its low cost allows prototyping at a higher speed than with traditional methods, thus enabling the rapid adaptation of different types of production in extraordinary cases, or the development of an industry focused on customised production at a more affordable cost.

This technology, although it is currently working under the same bases as from the beginning (layer printing and STL formats) is under constant research to develop new post-production methods and materials to print, such as 3D printing of food [27], organ printing [28], development of fibre-reinforced polymers [29] and non-planar slicers [30].

2.5.1 SLA vs FDM

There are many 3D printing technologies, each with its own peculiarities, advantages, and disadvantages. The focus is going to be on the differences between Fused Deposition Modelling (*FDM*) and Stereolithography (*SLA*), which are the two available options in the lab.

FDM works by melting a polymer filament and extruding it along the base. This is done following the design. Once the layer is finished, the extruder is raised and prints the new layer on top of the previous one. The process is repeated until all the layers are fused together. It is widely used due to its low cost, good resolution, and relatively fast print depending on the shape and infill.

SLA technology is very different. Instead of depositing material, the printing bed is submerged in a tank full of photo-reactive polymer resin. With the help of a laser or ultraviolet light, the resin is cured and solidified layer by layer. Once the print is done, the piece is removed and put in a washer station where the piece is submerged in isopropyl alcohol and a motor spins the isopropyl around removing the surplus resin. Finally, as the piece is not completely solid yet, a cure station with UV light is required to harden the piece. With *SLA* you can achieve more precise and stronger prints, but machines are more expensive. It also takes more space as you need 3 stations, takes longer to have a final piece as the laser needs to hit each layer for a fixed amount of time, and you must be present when the print ends to do the washing and curing.

Now, research was done to use the most suitable materials for each task.

For the *FDM* printer there is a tone of filaments available in the market with many different properties. Made of different polymers, flexible, tough, conductive ... Polylactic acid (*PLA*) is a great option as it is widely available, comes from renewable sources (bioplastics), has good layer adhesion, low melting point and high strength. Furthermore, for more robust, durable, and final prototypes, the proprietary *MakerBot Tough* filament is a great option. Being 2x more resistant to impact than *ABS* and it does not require a heated bed to print [31]

For the *SLA* printer, the most appropriate resin between the supported ones is the *Though 2000*. As power tools are used in hostile environments, such as construction sites, it requires a very long-lasting resin. *Though 2000* is the strongest and stiffest resin from *FromLabs*, creating sturdy pieces that do not bend easily with minimal deflection [32].

2.6 Printed Circuit Boards

Printed circuit boards, also known as printed wiring boards, are used to construct electrical devices. When building an electronic device, a PCB performs two functions: it acts as a mounting surface for the components and as a conduit for electrical connections be-

tween them. The board is usually manufactured using layers of resin, copper and fiberglass. The copper layer makes the connections while the other two components provide electrical insulation to avoid short-cuts and provide thickness to the board. With technology evolution, PCBs can go from a single layer of copper up to 20+ layers [33]. The complexity and production cost simultaneously grows alongside the number of layers. More layers allow more compact and dense boards; however components can only be placed in the outer layers. This forces boards to be bigger to include all the modules which simultaneously reduces the amount of layers needed.

Flexible *PCBs* also exist for applications that require them such as moving components, connectors, straps [34], shoes [35] or sensors [36].

Nowadays PCB production is more accessible than ever. Anyone with a computer can design the board and send it to one of the many PCB manufacturers to get it printed at affordable prices. In the past, the minimun order was to high for personal or small projects but now it can be as low as 5 boards [37]. Most of the companies also provide the assembly of components. This way projects can jump in quality eliminating breadboards or hand-soldered protoboards.

3 State of the art

General-purpose trackers are versatile tools that are starting to become massively used. They are a valuable investment for businesses, families, and individuals who need to track the location of people, packages or any kind of assets. One of the best-known tracking devices is AirTags [38] from Apple. These are *BLE* based devices which provide a great tracking experience by sending periodically beacons from the tag. These are captured by one of the millions of active *Find My network* Apple devices and it will automatically publish the position of the detected tag to the network. When the owner is nearby the tag, the distance to the AirTag and the direction to head in is displayed, it is also possible to make it beep. It is powered by a button cell which provides an approximate 1-year autonomy. This system works worldwide but requires the user to own an Apple device. Furthermore, as it has to be close to a *Find My network* device to obtain a position, in the countryside or remote areas is not that effective.

For this reason, other GPS-based alternatives have a spot in the market. Providing a precise location anywhere outdoors which is sent through *GSM*. These alternatives drain more power and require a SIM card with a plan to send the position of the tracker. They are bulkier than the AirTag and hence used to track bigger things.

These general-purpose trackers do their job, but how can they be implemented with power tools to track and monitor them? The main power tool manufacturers have their own approaches. An overview will be presented of the main tracking solutions.

Milwaukee

Milwaukee has the ONE-KEY™ network and app for monitoring, managing, and tracking tools. They have the Milwaukee Tick™ Tag (Figure 8) which works with Bluetooth and is powered by a cell battery, like the *AirTag*, but it can be attached in more robust ways. The network used is built from the user community of the app. But they need to have the app running on the background of their phone. This is not very reliable as the number of users is low, not everyone wants to get their battery drained, and the 30 meters range where the user has to be in order to capture the signal is short distance.

They also sell ONE-KEY ™ enabled tools (i.e. Figure 9). The hardware is built into the tool and cannot be manipulated. It enables behaviour configuration, RPM limitations, battery level and blocking the tool under command or if it goes out of range. It even works without a battery plugged (no information found about how long). The drawbacks are that the tools are high-end series, they rely on their poor ONE-KEY ™ network and no audible or visible feedback can be produced by the tool to find it.

Figure 8: Milwaukee Tick™ Tag [39]

Figure 9: Milwauke ONE-KEY ™ enabled tool [40]

DeWalt

DeWalt's solution is called TOOL CONNECT™ and the system encompasses accessories, tools, an app and a web portal. It provides managing inventory, customising settings and tracking assets. The connections are also Bluetooth based and rely on their own app community network with the drawbacks that it brings. They have the Tool CONNECT™ Tag attachable to any asset water and dust resistant (see Figure 10). The TOOL CONNECT™ Chip that merges inside the tool (see Figure 11). Both products are powered by one-use coin cell batteries.

Figure 10: Milwaukee Tick™ Tag [41]

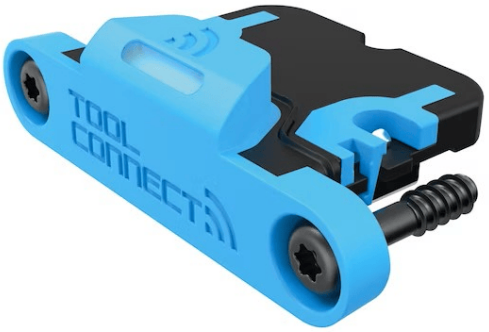


Figure 11: DeWalt TOOL CONNECT™ Chip [42]

They also sell power tools and batteries with TOOL CONNECT™ already integrated (see Figures 12, 13) which can be deactivated in case of theft. Moreover, DeWalt has a TOOL CONNECT™ connector for non-native supported tools. It consists of a permanent lock-in add-on in between the tool and the battery. It has a rechargeable coin cell battery that recharges from the tool's battery and enables connectivity to older tools (see Figure 14). Finally, to expand wireless capabilities on the job site and get better track of the assets, they sell a gateway (see Figure 15). All the products include a blue LED as visible feedback.

Figure 12: Milwaukee Tick™ enabled tool [43]

Figure 13: DeWalt TOOL CONNECT™ battery [44]

Figure 14: DeWalt TOOL CONNECT™ Connector [45]

Figure 15: DeWalt TOOL CONNECT™ Gateway [46]

Bosch

The connectivity proposed by Bosch is called *Simply.Connected.* It connects power or measuring tools through Bluetooth to the *Bosch Toolbox App.* For power tools, it provides KikBack adjustment, LED configuration, battery status, enables or disables the precision clutch and quick access to manuals. The connectivity can be already built-in the tool or plug-in with an additional module (see Figure 16). The connectivity module *GCY 30-4*, is powered by a coin cell battery and is easily installed in the tool's grip. Which also makes it effortless for thieves to remove. For the measuring tools, the values, IR images and configuration can be seen and modified in the app.

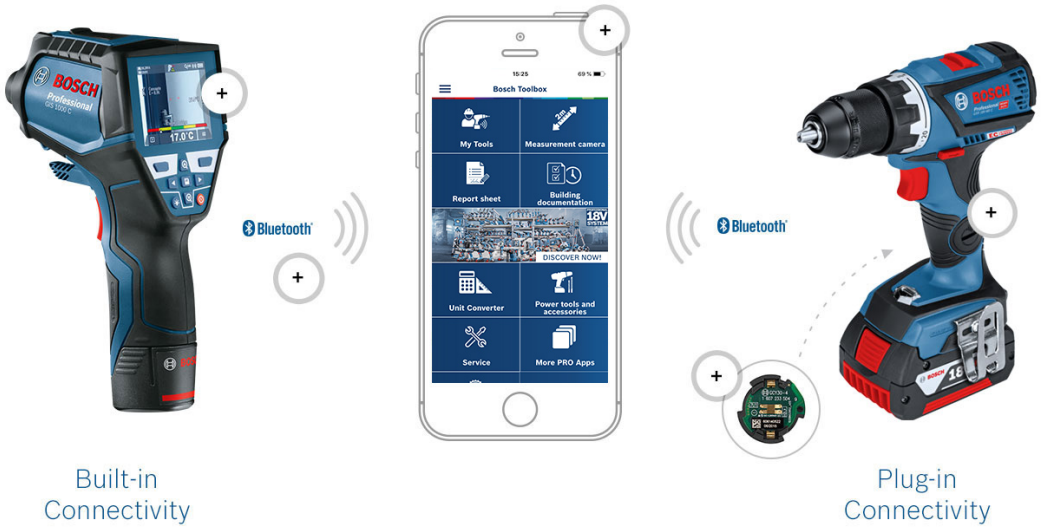


Figure 16: Bosch solution

Now that all the options are known, we can tell all the problems found in the existing options.

- Some are not easily integrated into a tool
- Simple tags do not include monitoring information of the tool
- Some can be easily manipulated and spotted
- Some are powered by one-use batteries
- Vendor lock
- Poor and unreliable community networks
- Not available for all the series
- Poor or no feedback from the tracker

The proposed add-on aims to unify the advantages of the existing devices and solve their weak points. To achieve this, the theoretical overview previously done will be used to select the appropriate technologies. First, a reliable LPWAN network will be implemented instead of a community-based network. The add-on uses it to communicate the location and status with the backend anywhere and anytime. In the backend the data can be managed and many applications can be developed. The design will be similar to Figure 14 implementing also a backup battery. The add-on is meant to work with every brand and tool series (structural modifications will be required but not internal), allowing clients to have a unified system for all the brands. Accurate tracking is a must both indoors and outdoors so WiPS and GNNS can be used for this task. Besides, it can enable monitoring functionalities. Furthermore,

hearing user's complaints about the existing solution is really useful. From reviews, it was noticed the lack of effective feedback from the trackers to help them find tools on job sites. A loud buzzer could be very useful for this, as sometimes they can be out of sight inside a case or hidden behind some materials.

In the next section, the hardware and functionality requirements will be listed to have a goal to reach.

4 Requirements

For this first prototype project, the resulting add-on has to fulfil these requirements:

- Retrieve a precise position
- Work indoors and outdoors
- Low power consumption
- Have a good communication coverage
- Allow normal functionalities of the tool
- Structure has to be 3D printed
- Easy to install and remove for convenience while testing
- Has to be resistant
- No electronic components are exposed to the outside
- Cheap production and maintenance cost
- A versatile backend that enables downlink communications
- Sonorous response to help locate the tool when is lost
- Tool's battery as the main power source (range 20.5V - 16V)
- A backup battery that lasts at least 2h
- Some monitoring of the tool
- Work in Europe
- Display the information in an online dashboard

5 Add-on design

5.1 Hardware selection

Before designing in detail our prototype, the main hardware should be chosen. Taking into consideration the required space for the modules inside the add-on. The ideal would be to do it as smaller and power efficiently as possible, but this is usually more expensive, requires a complex *PCB* design, and low-level programming for embedded systems. For this first prototype, simpler modules will be used, leaving this option for the development of the final commercial product.

5.1.1 Microcontroller

The microcontroller used must incorporate a *WiFi* module, and communicate through *LoRa*, I/O ports, and power pins. During the research to find a suiting, cheap and small option. Two main manufacturers were found with similar products. HELTEC with the *WiFi LoRa 32 (V3)* [47] and Lilygo with *TTGO LoRa32 V1.3* [48]. The Lilygo solution was selected for its lower price, same functionalities and already existing hardware at the lab.

More about the*Lilygo TTGO LoRa32 V1.3*. It is based on an ESP32-D0WDQ6 chip [49], paired with an SX1276 [50] *LoRaWAN* module, 0.96 Inch *OLED* Screen, a 12-bit SAR ADC, JST HXP-2 connector for an external battery, built-in *TP4054 LiPo* charger, supports *WiFi* and *BLE*. The 36 available pins provide versatility and many functionalities. Including 16 ADC measurement channels, *UART*, *SPI* and *I2C* pins (for more detail see Figure 17). The voltage input of the *LiPo* goes from 3.7V to 4.2V including a smart power management with over-charge and over-discharge protection. It can conveniently be programmed using the Arduino IDE. All of this makes it a great device for prototyping and easy debugging. However, for the final product, it is not the best option, as it is quite big (see Figure 18) and not very energy efficient due to not needed features.

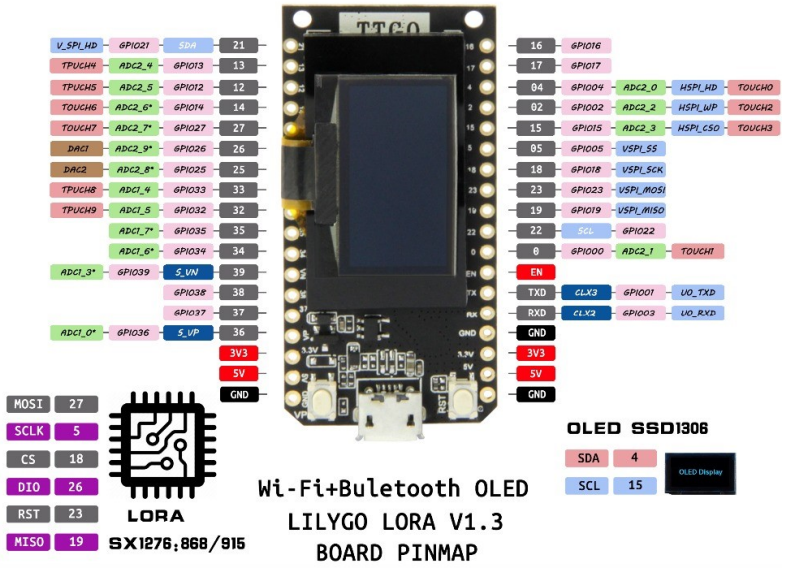


Figure 17: LILYGO-TTGO-V1.3 PINMAP

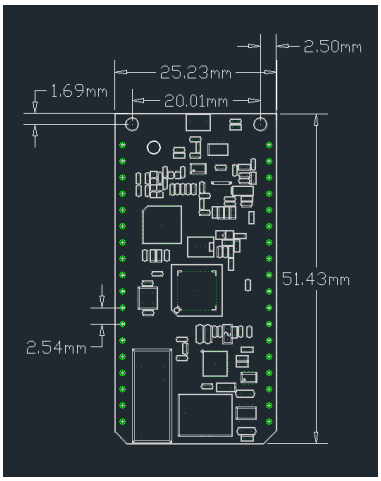


Figure 18: LILYGO-TTGO-V1.3 measurements

The chosen LoRa antenna is not included with the board. The included antenna has an SMA connector and requires an MHF adaptor, taking up space. For an easier placement, smaller format and no need for adaptors, the Molex ISM 105262 [51] flexible antenna was selected. It also includes an adhesive side for quick installation.

One drawback is that Datasheets are almost non-existent for the *Lilygo* modules. Only very basic information is provided. But it is also true that testing and measuring the power consumption is required to know exactly the capabilities of our prototype and power consumption. With these measurements, the code can be optimized, know how long will it last on standby and allow to know better the setup.

5.1.2 GNSS module

When monitoring the tool and no WiFi signals had been found, the *GNSS* module is used to obtain a precise location. Out of the already available modules, one based on Ublox NEO-7N-0-002 [52] was selected. The exact module that was in the lab could not be found anywhere online available for purchase. From this blog [53] the information was extracted. The main advantages are the working power range from 3.3V to 5V, and its compact design with a 1575R-A Y [54] antenna soldered on top of it. It has the connection for an external SMA antenna that can be soldered. Another strong point it the low power consumption of 40mA (according to [53]) and the high sensitivity that provides. It supports *GPS*, *GLONASS*, *QZSS* and *SBAS* [52]. To communicate with the module, it has a UART and a USB interface.

The fixing times announced by the manufacturer can be seen in Table **??**. Further testing will be done in Section 9.

		NEO-7N
*Time-To-First-Fix	**Cold Start**	29s
	Warm Start	28s
	Hot Start	1s
	Aided Starts	5s

Table 2: Manufacturer time to fix GPS satellites ∗Time-To-First-Fix (all satellites at -130 dBm) [52]

5.1.3 *DC-DC* regulator

The add-on hardware has to work in the usual electronic range of 3.3 - 5 V. In principle, 3.3V is the best option if all the modules can work with that voltage. This is because the current is lower with 3.3V than with 5V if the load is the same. Furthermore, the internal regulators of each module will not need to work, increasing efficiency by only using one better regulator. However, adding a backup battery forced the voltage output to be 5V so the microcontroller can charge the battery. As using resistors is very low efficient, transforming the exceeded voltage into heat, and we are running on a battery it was discarded. A great alternative is including a DC-DC step-down regulator to efficiently reduce the voltage.

When reducing a lot the voltage, the buck regulator [55] is more efficient than the linear regulators. This is because buck converters reduce the voltage by switching on and off a transistor, this is done thousands of times per second to chop the signal and achieve the desired V_{out}. The main problem they have is the generation of ripple by the switching. It can be reduced with the help of capacitors or new design variations [56].

The output of the power tool battery (input of the regulator) can be up to 20.5 V when fully charged. So the desired regulator has to be able to handle at least that amount of voltage. The V_{out} has to be 5V fixed. Also, the expected draining of the whole add-on is not very high, so it will require an output current of 500mA in to be able to handle peaks when sending or using the GPS. It has to be small, efficient and easy to implement in the custom PCB.

An online comparison was done and finally the TPSM84205EAB [57] was selected. It is a DC-DC step-down SWIFT^{TM} [58] regulator. It is a proprietary implementation of a buck regulator developed by Texas Instruments. This model can take up to 28V and has a fixed output voltage of 5V. The approximate dimensions are 10 x 11 x 8 mm and it has a standard TO-220 footprint with 3-pins: V_{in}, GND and V_{out}. The efficiency depends on the input voltage and for our use case is around 88% as can be seen in Figure 19[57]. It can provide up to 1.5A without the need for a heatsink.

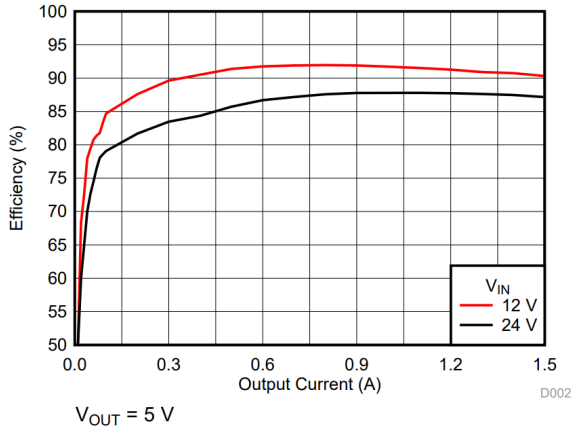


Figure 19: Efficiency vs Output Current TPSM84205EAB [57]

5.1.4 LiPo battery

One-cell LiPo batteries were selected due to their high availability, power density and compatibility with the microcontroller input voltage range and charging circuit.

Two options were available at the lab with similar dimensions but no capacity, weight or thickness. One is $1800mAh$ [59] and 36g whereas the other is $450mAh$ and weights 10g. The election depends a lot on the current drained by the microcontroller plus the GPS and the available space. To see how long will it last running on the battery, the consumption is measured in 9.3.

With those measurements, we can do the calculations to check if the backup battery lasts at least 2h and fulfils the established requirements. Using the *LMIC standby* consumption from Table **??**, the autonomy provided by the $450mAh$ battery would be approximate $BatteryLife = \frac{0.45}{0{,}0431} = 10.44h$ and for the $1800mAh$ battery $BatteryLife = \frac{1.8}{0{,}0431} = 41.76h$. With these values, the $450mAh$ battery was selected for providing the required autonomy and being smaller and lightweight.

Disclaimer about the calculations. The *LMIC standby* value takes into consideration the regulator consumption (which the battery does not power). Furthermore, is assumed as an average consumption for the add-on, not considering consumption increases when using GPS, WiFi or LoRa. Despite everything, it helps to have a general idea and allows us to choose between both battery options as both options exceed by far the requirements.

5.2 Printed Circuit Board

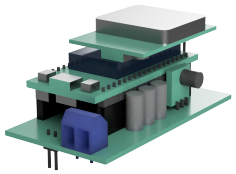


Figure 20: 3D design of the PCB with the modules to check the size

5.2.1 PCB design

The chosen software for designing the *PCB* is *Eagle*. Is also from the *Autodesk* suite, but has more powerful tools for *PCB* design than the included in Fusion 360. For this first prototype, a simple *PCB* containing all the required modules and hardware from the Schematic 21 was designed. First of all, a 3D design was done in Fusion 360 measuring each component to see how to place them in the PCB and fit inside the add-on (see Figure 20). By importing it as a *Derive* to the main add-on design, the parts could be quickly rearranged and get automatically updated in the initial design. The selected vertical stack mount is in order to optimize the available space and try to reduce as much as possible the add-on height.

Once the placement was clear, the schematic from Figure 21 was portrayed in the footprint from Figures 22 and 23. For some components such as the microcontroller or the GNSS module the library was created whereas, for the rest, existing libraries were used. Ratsnets were used in order to create bigger traces in high current connections and to create common GND on both sides of the *PCB*.

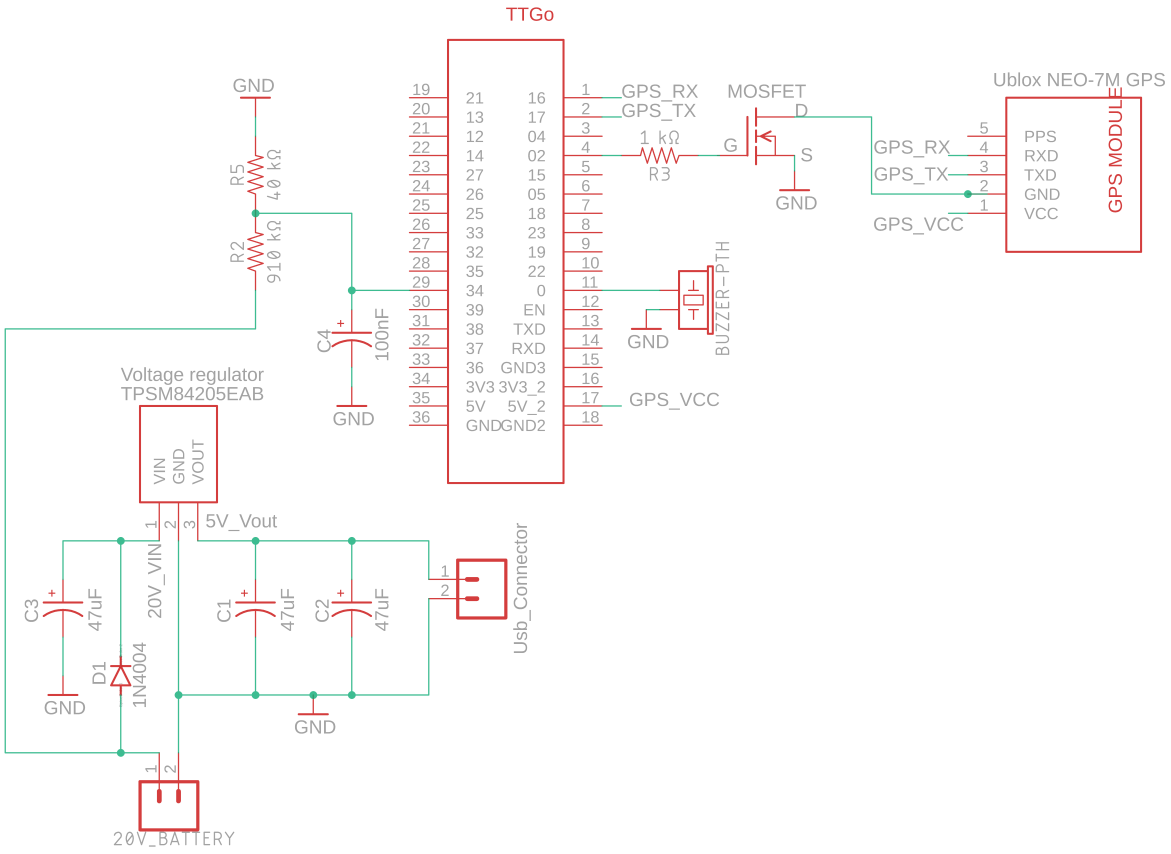


Figure 21: PCB Schematic

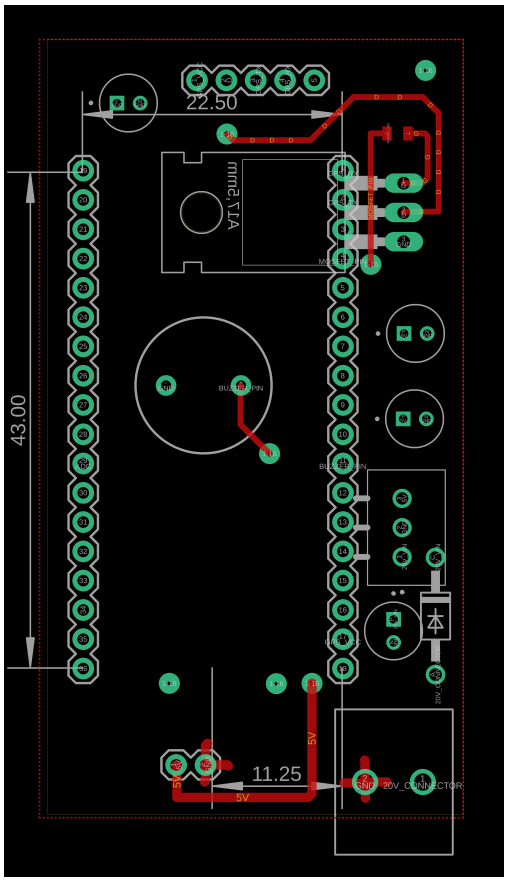


Figure 22: Footprint top layer

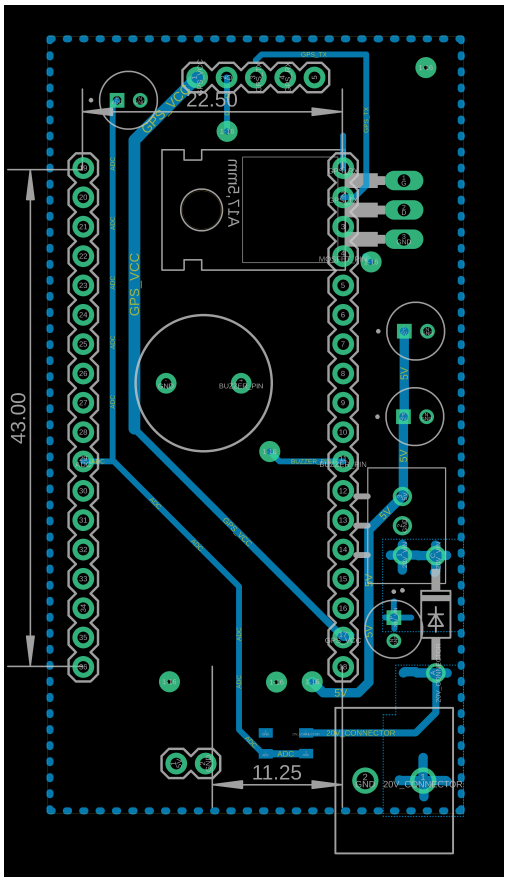


Figure 23: Footprint bottom layer

Now a walk-through of all the components and traces will be done explaining all the connections and elements.

The voltage input goes into the PCB from a power strip through a diode until the DC-DC step-down regulator *Vin*, where a capacitor is used to store energy and avoid sudden shortages. The diode prevents the capacitor to help the main battery when a voltage drop occurs while using the power tool. This way it will only support the regulator and not the tool. Furthermore, the diode forward voltage of up to 1V [60] is even beneficial for the regulator, having a lower input voltage and increasing efficiency (see Figure 19)

Capacitors are included at the V_{in} and V_{out} of the regulator following a pi filter structure in order to avoid spikes. This is the recommended usage according to the manufacturer [57] from where the values of the capacitors were also obtained.

Because of the diode voltage drop, the voltage divider must get the voltage straight from the source to accurately monitor the battery. To establish the values of the voltage divider resistors, a couple of things have to be taken into account. The working range of the power tool battery is from 16V to 20.5V. As well as the working voltage of ESP32-D0WDQ6 included ADC. Which has different errors according to the voltage range (see the ESP32-

D0WDQ6 data sheet [49]). The lowest error is achieved in the range of 100 $\sim$ 950 mV. Besides, according to the Ohms law, for the same voltage the greater the resistance, the less current. So big resistor values were sought but taking into account the available resistors in the lab.

The selected values are R1 910 kΩ and R2 39kΩ. After doing the calculations with the maximum V_{in} case, we obtain a valid value leaving a small margin of 100 mV:
$V_{out} = V_{in}(max) * \left(\frac{R2}{R1+R2}\right) = 20.5 \left(\frac{39000}{949000}\right) = 0.8425V$

The voltage divider drains: $I = \frac{20.5}{949000} = 21.6\mu A$

In order to have more stable readings, a small 100 pF capacitor was included in parallel to R2. The resistors used are from the series 0805 which have a small format while being easy to manually solder them. Another consideration is which ADC channel to use. Channel 2 is used for WiFi [49], so channel 1 will be used instead to avoid conflicts while scanning for WiFi signals and reading the voltage.

The 5V output then goes to a micro USB connector from where the microcontroller is powered. By checking the Lilygo TTGO schematic (see Appendix A), the only way to take advantage of the built-in TP4054 LiPo charger is to power it from the USB. Otherwise, if the microcontroller is powered from both a 5V pin and the battery, the protective fuse will burn as happened while testing.

The GPS module is connected to the 5V pin of the microcontroller. Therefore if the regulator does not get power because the user disconnects the tool's battery, the module can still be powered by the backup LiPo. RX and TX are connected to the microcontroller to establish the UART communication. And the GND is switched by the mosfet in order to turn it on and off on command.

The passive buzzer is connected to GND and the other leg to a pin of the microcontroller. As the buzzer does not include an internal oscillator, the microcontroller has to create an AC audio signal to make it sound.

5.3 Add-on behaviour

This first prototype of the add-on will follow a simple workflow to test all the functionalities. This behaviour is defined in Figure 24 where the microcontroller first sends a *Join Request* message to the Helium network. If there is no coverage, it will go to sleep and try later, it is a waste of energy trying to obtain a more precise location and not being able to send it to the backend. Otherwise, the add-on will receive a *Join Accept* package. Then the second Rx window will be open and perhaps the GPS or Buzzer could be turned on. If no message is received, the WiFi scanning will start. But if a downlink package is received, first

the buzzer will be actuated, afterwards if the GPS had been turned on directly use it, and if not, back to the WiFi scan.

If the scanning is successful and at least two WiFi signals are found, it will be parsed and sent to the *LoRaWAN* network. Opposite, if no signals are detected, the GPS module will be turned on for 70 seconds (see tests in Section 2.2) and a precise location obtained ready to be sent. If the GPS module wasn't able to fix satellites, no precise location was found either with the WiFi nor the GPS so the approximate location of the *Helium* gateway that received the *Join Request* will be used. Therefore, no useful information can be sent from the add-on, thus the microcontroller will go to sleep. As the device is LoRaWAN Class A, after transmitting the GPS information two receiving windows will be open. If there is a downlink message, the procedure will be the same as explained before. Finally, when all the tasks are done, the microcontroller will go into sleep/ low consumption mode to save energy.

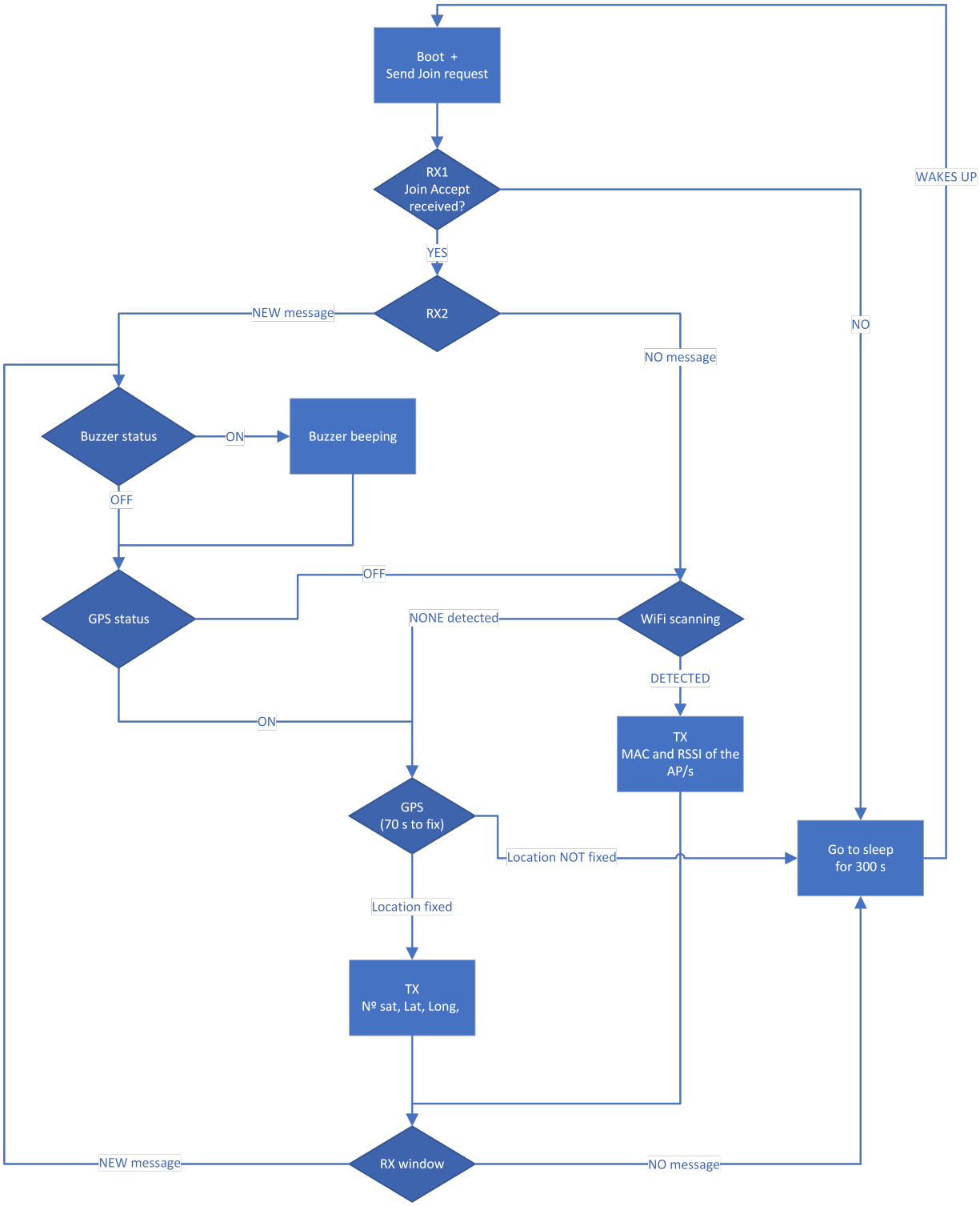


Figure 24: Flowchart with the basic behaviour

This is the general diagram, more complex and adapted diagrams will be required for different implementations and companies.

5.4 LoRaWAN network provider

The *LoRaWAN* network provider is another thing to select. The two main options are The Things Network (*TTN*) or Helium. *TTN* as a free public community network, has a fair use policy which limits the uplink airtime to 30 seconds per day per node and the downlink messages to 10 per day per node. On the other hand, if you use a private network, these limits do not apply, but you still have to be compliant with the governmental and *LoRaWAN* limits. As the idea is to use the already existing network, the fair use policy is quite limiting.

The other option is the Helium ecosystem. It is a global, distributed network of hotspots that create public, long-range wireless coverage for IoT devices. Its hundreds of thousands of hotspots deployed by a global community, provide access to the largest decentralized wireless network in the world. Helium gives 250 free *DC* credits for testing and then charges $0.00001 per 24-byte (more in Section 6.5). Also, the location of the gateway is provided with the message allowing a first unprecise location in case the device is only able to send the *Join Request* message.

5.5 Custom communication protocol

A custom communication protocol over the LoRaWAN connection was created for the project. The aim of the protocol is to send and receive the information with the smallest possible packet sizes. Therefore the minimum amount of bytes will be transmitted, taking less air time and power. This translates into fewer duty-cycle restrictions, minimum power usage and billing, as Helium charges per byte transmitted.

Uplink packages

There are two main types of upload messages depending on the availability of *WiFi* signals. If at least two *APs* are found, the message will contain the *MAC* address without spacers, the *RSSI* values without sign and the battery percentage. This way, useless information is avoided. *RSSI* values can only be negative or zero, and *MAC* addresses always have the same size. By putting more effort into the backend decoder, it saves plenty of repetitive information.

If the micro-controller finds more WiFi *APs*, they are first sorted by signal strength (the more significant the better) and only the three top signals will be sent (see testing in 9.6). Each signal will follow the previously explained technique and be appended one after the other with the battery percentage always added at the end, independently of the number of *APs*. The structure can be seen in Figure 25. The two possible message sizes are 29, and 43 bytes corresponding to sending 2 or 3 WiFi signals (see why in Section 9.6).

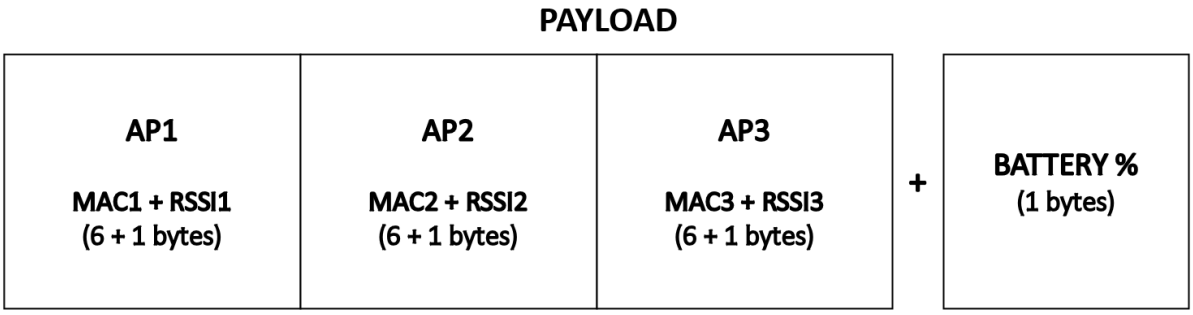


Figure 25: Structure of the payload when sending an uplink message with WiFi data

In case the *WiFi* scan is unsuccessful or only one *AP* is found. The *GNSS* module will be used to obtain a precise location. Once the module has run for 70 s and fixed at least 3 satellites, the latitude and longitude are retrieved and stored using 6 decimals. This way our accuracy can be up to 111 mm, enough for this use case. The altitude is omitted for now, but it can be reconsidered in the future. The number of fixed satellites is also included in the message to determine the precision of the data. The package shape can be seen in Figure 27. Packet size is fixed to 21 bytes, taking into account the ranges for each of the variables explained in Section 2.2.

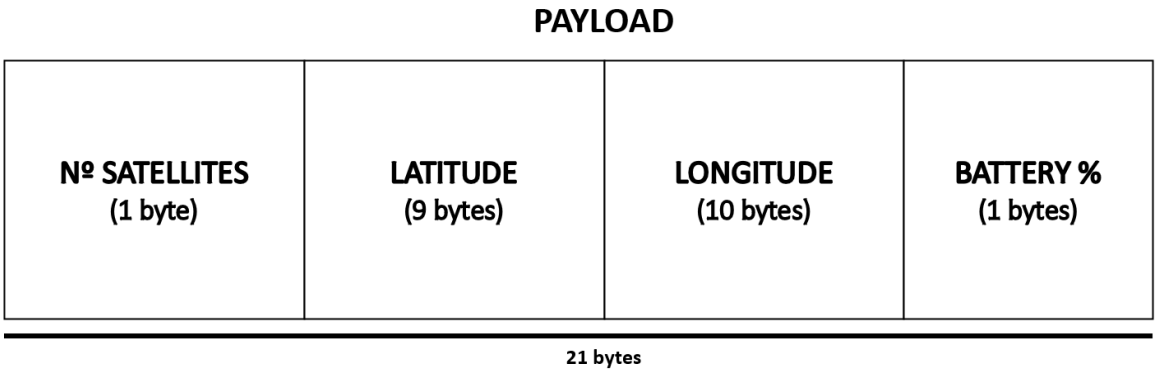


Figure 26: Structure of the payload when sending an uplink message with GPS data

Downlink package

There is only one simple downlink package to communicate with the microcontroller. Remember that as the device is *Class A* downlink communication is only available after a transmission. This package consists of a single byte that can take 4 different values. Each value turns ON and OFF the GPS module and/or the buzzer. All the combinations can be seen in Table **??**.

Value	GPS	Buzzer
0	OFF	OFF
1	ON	OFF
2	OFF	ON
3	ON	ON

Table 3: Symbol encoding used for the downlink packet

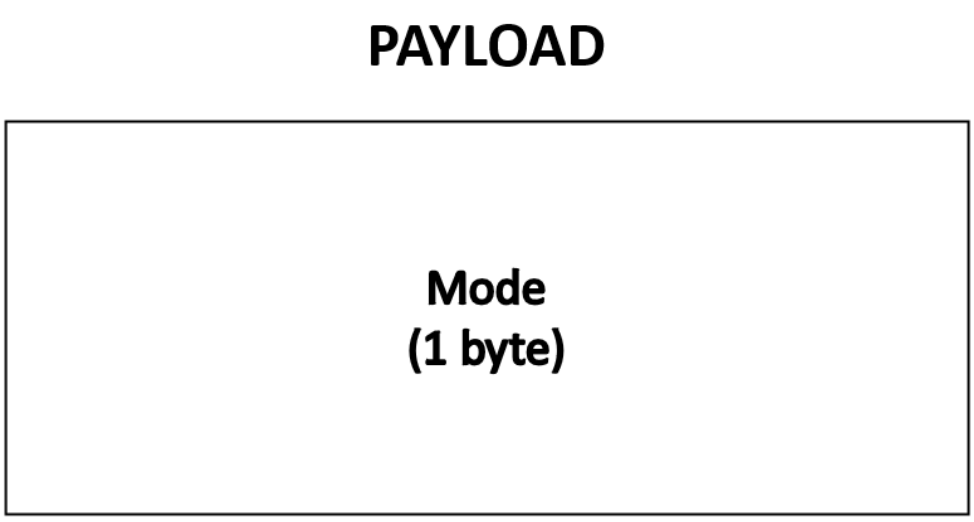


Figure 27: Structure of the payload when sending a downlink message to switch ON or OFF the GPS module and/or the Buzzer

5.6 3D design

The idea is to make an easy installation add-on similar to [45], it should use the existent connectors and mechanism of the power tool. So, the first step is to try to mimic the existing connections. There are two main connections, the one in the power tool 29 and the one in the battery 28.

Figure 28: Battery connector

Figure 29: Power tool connector

In 28 we can see how the battery has a 4-slot female electrical connection, but the global piece is a male connector that goes into 29. And vice versa, 29 is hollow and has 3 male metal connections (+20V, GND and thermal protection signal). The missing connector is only used for the balanced charging of the battery so when plugin it to the tool is omitted. From this starting point, getting to copy those shapes is the challenge and produce a 3D printed piece that fits correctly both connections.

To replicate them we have two main options, take measurements, or use a 3D scanner camera. As taking measurements is very time-consuming, and there is a *Revopoint Mini* available at the lab, this 3D scanner was used to quickly create a 3D model out of the real objects. In section 5.7 the process followed will be explained.

5.6.1 Revopoint Mini

This exact model is the most accurate from the brand *Revopoint* with a resolution of up to 0.05mm [61]. It is meant to scan small objects with high resolution. The pack includes a rotation table to make it easier to take pictures of the object from every angle. A small walk-through will be done below, for more details check the manual [62].

To begin scanning first start the *Revo Scan* software. Then mount the camera in the tripod and connect it to the *PC*. Afterwards, place the object in the rotation table and adjust the camera distance as in 30 with the help of the software.

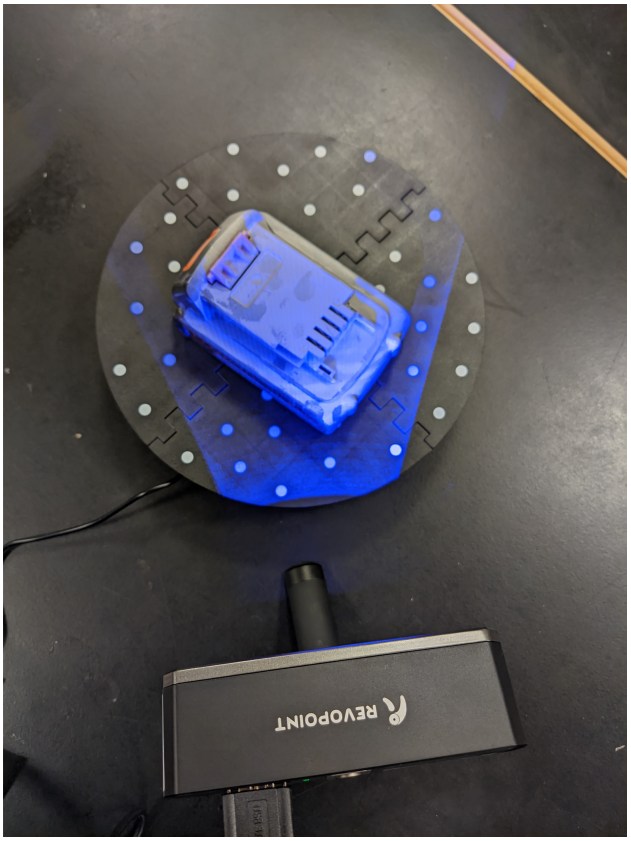


Figure 30: Scanner setup

With all the previous steps completed, now you can start a new project and select the correct configuration for the object scanned (i.e. look for tracking points or features, Color o No color, Accuracy, etc). Once is configured you can start the acquisition of pictures and the software will automatically detect points of the object, creating a cloud of dots. The software shows in blue the points captured of the whole object and in green the ones currently in sight of the camera as can be appreciated in 31. In our case, some parts are made from plastic and have a reflective surface. To avoid bad measurements due to reflections during the scan, a 3D scanning spray is applied. It creates a mate white coating and solves this issue.

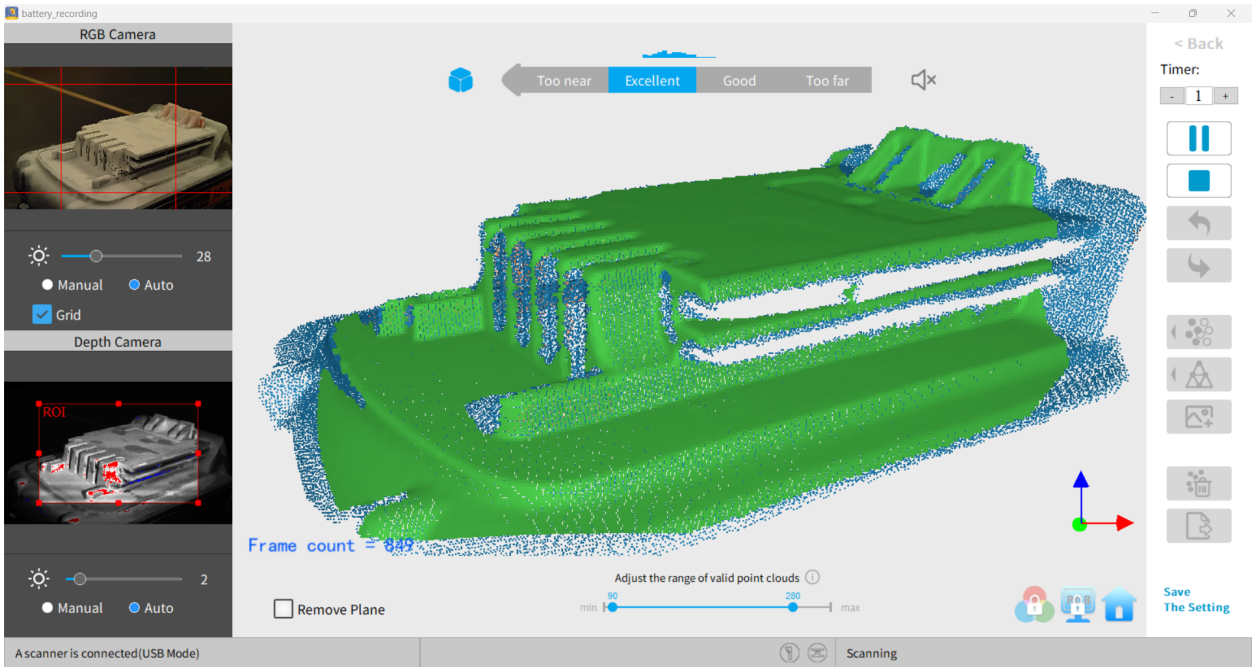


Figure 31: Scanning process using Revo Scan

When all the desired areas are scanned it is time to process the cloud of dots. First, duplicated points are merged to simplify and reduce the number of points. Later, they are connected in groups of three creating a mesh of triangles. This is a very computationally demanding process especially if there are lots of them or the specified quality is high.

With the help of another software from the same company, *Revopoint Studio*, a basic post-process can be done. Including some features like removing isolated groups, filling gaps, simplifying the mesh, and trimming or smoothing faces.

Finally, to start working with a more powerful 3D design software, the mesh must be exported into a compatible file type as stereolithography (*.stl*) or *.obj*.

5.6.2 Zeiss GOM Inspect

Following the workflow of [63], a very interesting software is *GOM Inspect by Zeiss*. This is a very complex program but only will be used to solve the problem of using a *3D* scanner, which lacks a coordinate system. The program has a feature which establishes the coordinate system by indicating 6 points of the object that match the 3 origin planes (*ZZZ-YY-X*) as shown in figure 32. It is very important to perfectly align the model in order to have parallel and perpendicular lines, making the following *3D* design steps easier.

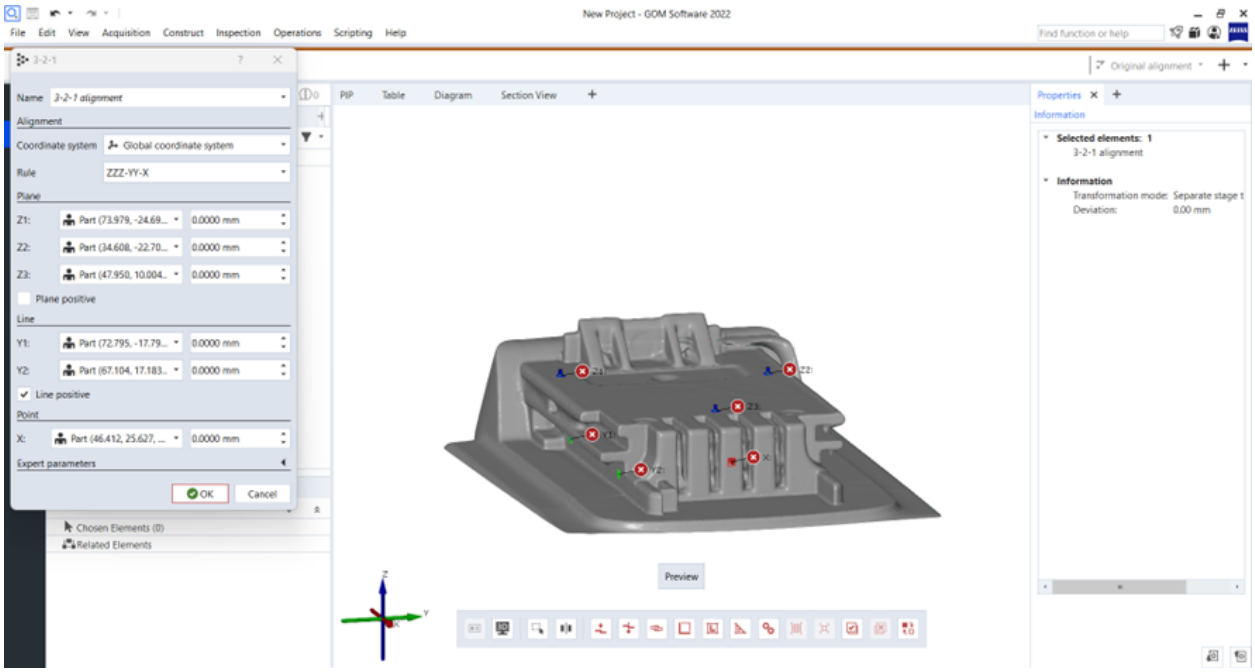


Figure 32: Establishing the coordinate system using GOM Inspect

5.6.3 Autodesk Fusion 360

Now with the resulting object, we import it to *Fusion 360*. The first noticeable thing is that the hollow parts are not perfect and have artefacts. Instead of trying to modify the mesh and in order to be able to precisely model the object a solid object is recommended. So the next step is to make a solid out of the triangle mesh. The automatic transformation turned out very badly. Therefore, as [63] suggests, doing sections of the mesh and then extruding the resulting sketch seems the best option. During this process, measurements with the caliper were taken to double-check.

Modelling the grip on the back of the battery was a challenging task, as it proved to be both time-consuming to create in sections and difficult to accurately measure. So it was opted for using only that part of the scan and use the automatic transformation to solid obtaining a decent result. Finally, merge the two solids. The *3D* scanned object mesh compared to the solid recreated can be seen in 33.

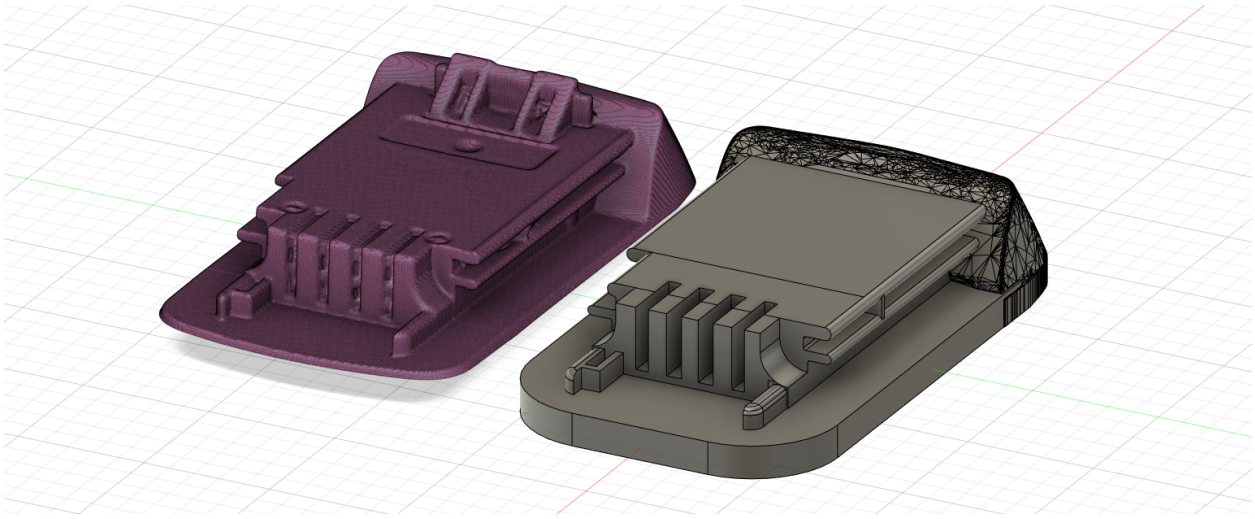


Figure 33: On the left the mesh obtained from the 3D scan and on the right the solid recreated out of it

For the other connector, the same process was attempted but the *3D* scan was very poor due to its very complex and hollow characteristics. So as a solution and to speed up the process, the existing male connector was subtracted from a solid cube. With some small modifications and giving tolerance, the obtained result was 34.

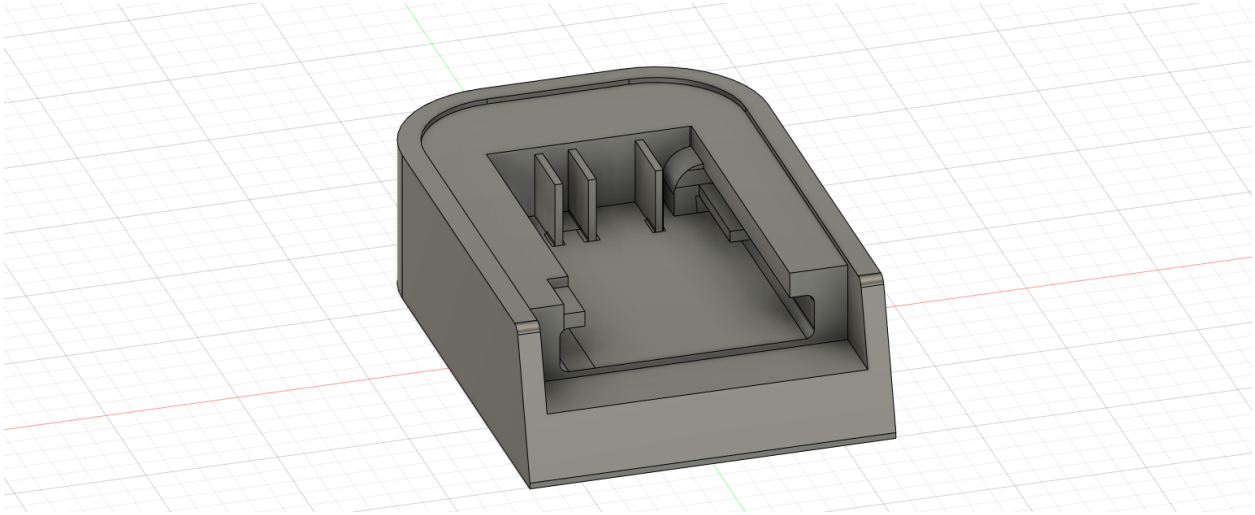


Figure 34: Power tool connector 3D design

The final idea is to stack/glue the pieces as in 51 creating a space between both pieces so all the required hardware can be enclosed in there. To be as accurate as possible when creating the hollow space basic shaped solids can be created using the measurements of the used hardware.

5.6.4 Modelling process summarised:

- 3D scan with Revopoint Mini, rotational table and applying the spray to avoid reflections.
- Revo scan to create the mesh and Revo studio to post-process the model (remove isolated parts, close gaps, create a mesh) and export as a .stl
- *GOM inspect* for establishing the correct axis of the object by selecting*ZZZ-YY-X* points in the respective axis.
- Using Autodesk Fusion 360
  - Further post-process
  - Make sections from the 3D scanned mesh to create sketches
  - Extrude them to obtain solid pieces
  - Check with caliper measurements and fine-tune them
  - Recreate hardware pieces required inside the add-on to create enough space for them
  - Export the resulting model and 3D print it

5.7 Prototype production

In order to have a real-world object and test if the modelling done is correct, *3D* printing was chosen for this task.

5.7.1 3D printers

The *FDM* printer used in the lab is the *MakerBot Replicator +* 35 with a printing area of 295 L x 195 W x 165 H (mm) and no heated bed [64].

Figure 35: MakerBot Replicator+

The *FDM* printer available at Skylab is the *Original Prusa i3 MK3S+* with a printing area of 250 L x 210 W x 210 H (mm) and no heated bed [65].

The *SLA* printer used in the lab is the *Formlabs Form 3* 36 with a printing area of 145 L x 145 W x 185 H (mm) [66], including the washing 37 and cure 38 stations.

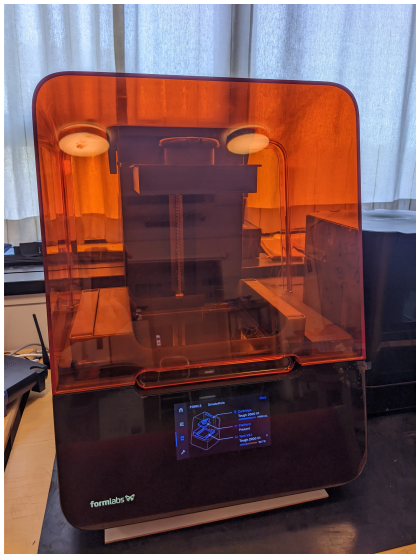


Figure 36: Form 3 printer

Figure 37: Washing station

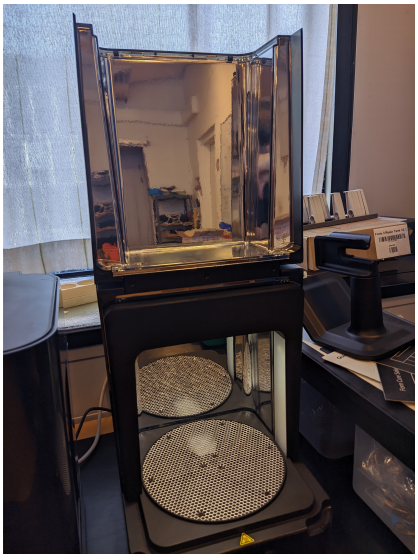


Figure 38: UV cure station

5.7.2 Material for printing

After taking into consideration the previous analysis of the two available printers, it was decided to use the *FDMs* for fast prototyping during the design phase and leave the *SLA* for the final product.

The fist prototypes were printed and turned out nice. In figure 39 we can check the battery connector and in figure 40 the power tool one.

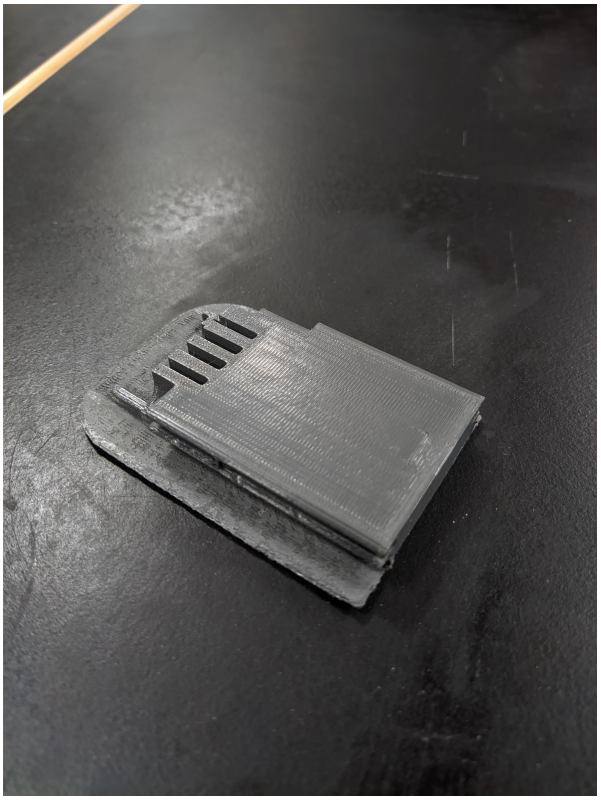


Figure 39: First battery connector prototype

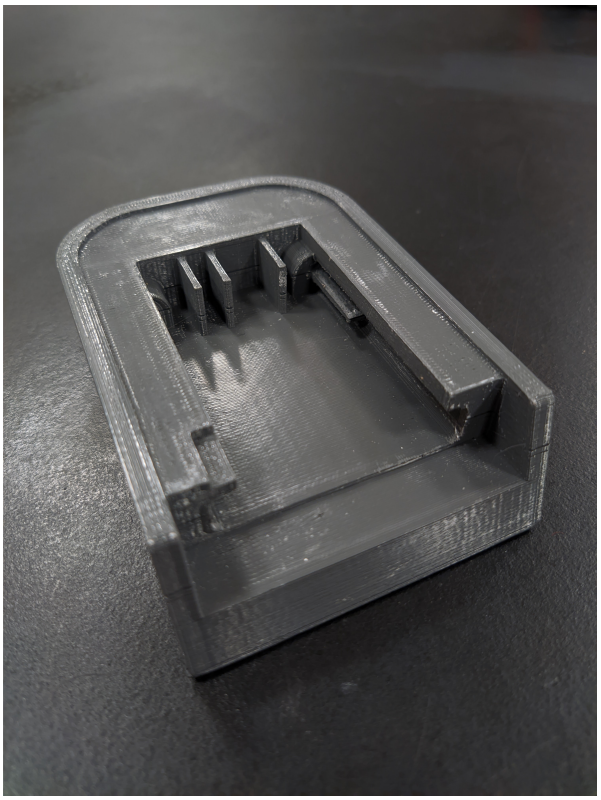


Figure 40: First power tool connector prototype

5.7.3 PCB production

The first set-up of the PCB design was done on a breadboard to check if everything worked fine and if the schematic could be used or needed some adjustments. In the beginning, when the tool was under heavy use, there was a voltage drop in the circuit big enough for the microcontroller to reboot. By changing the V_{in} capacitor to a bigger one and including a diode so the current from it only could go to the regulator, the problem was solved. Furthermore, with the backup battery, no more reboots have been experienced.

Once the design has been created and revised in Eagle, the production of the PCB can start. In this case, it is going to take place at DTU's IoT lab with the help of the research

assistant *Anas Mohamed Al Shalyan*. We decided not to order the product from an external company and instead get to experience and know the real process. It is also because of the faster production of the prototype. For example, *JLCPCB* [37] a big PCB manufacturer, takes at least 1 week to deliver the boards. Has a minimum order of 5 units and the shipping cost is more than double the cost of the PCB. Besides is a simple two-layer *PCB* that can be hand manufactured. The process followed was:

First, the negatives of each side of the board are printed on a translucid Laser Star artwork film sheet. Afterwards, they are aligned and glued with tape to a small piece of *PCB* that will help with the alignment, taking into consideration the thickness of the *PCB* (see Figure 41). Next step, in a dark room, peel off the protective plastic from the virgin board, put it in between the prints and UV-cure it for 75s. During this time, the thin layer of photosensitive protector will be removed from the areas exposed to the light. Immediately afterwards, get the board in the *BEL FECL3 PLETFJ* developer where the areas affected by the light will get lighter and the ones that were not, are darker 42. Once the difference is noticeable in the board, wash it with water and get the board into the etching machine. Here, the *PCB* fine etch crystals mixed with water are warmed and spread with bubbles, helping the process and making it more even. Approximately in 6 minutes it should be etched and the undesired copper removed (see Figure 43).

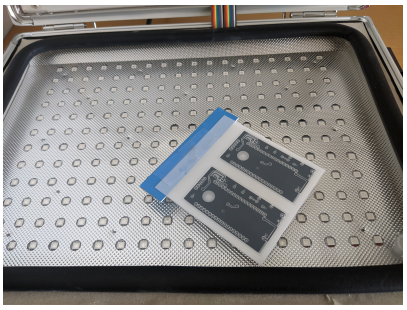


Figure 41: Negatives of the board resting inside the UV-curing box

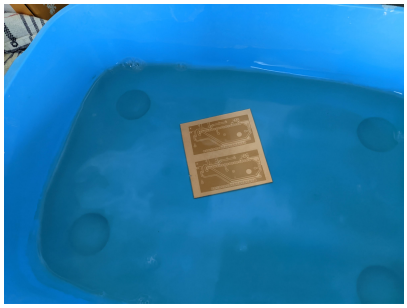


Figure 42: PCB in the developer

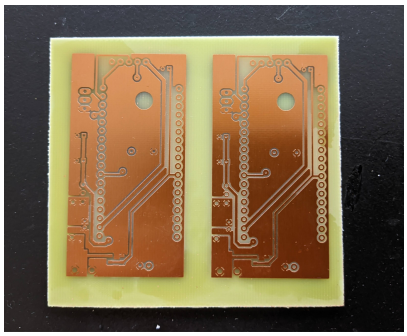


Figure 43: PCB after been etched

Afterwards, the board is cleaned with acetone to remove the rust layer left from the etching. Then, with a guillotine cut out the desired part and drill the holes with different drilling bits according to the size of each component pins. With all the holes done, the vias are installed. Afterwards, each component is placed and weld to the board. Finally, before connecting the power, testing with a multimeter must be done to see that all the welds are correct, there are no shortcuts and each connection has continuity.

The resulting assembled PCB can be seen in Figure 44.

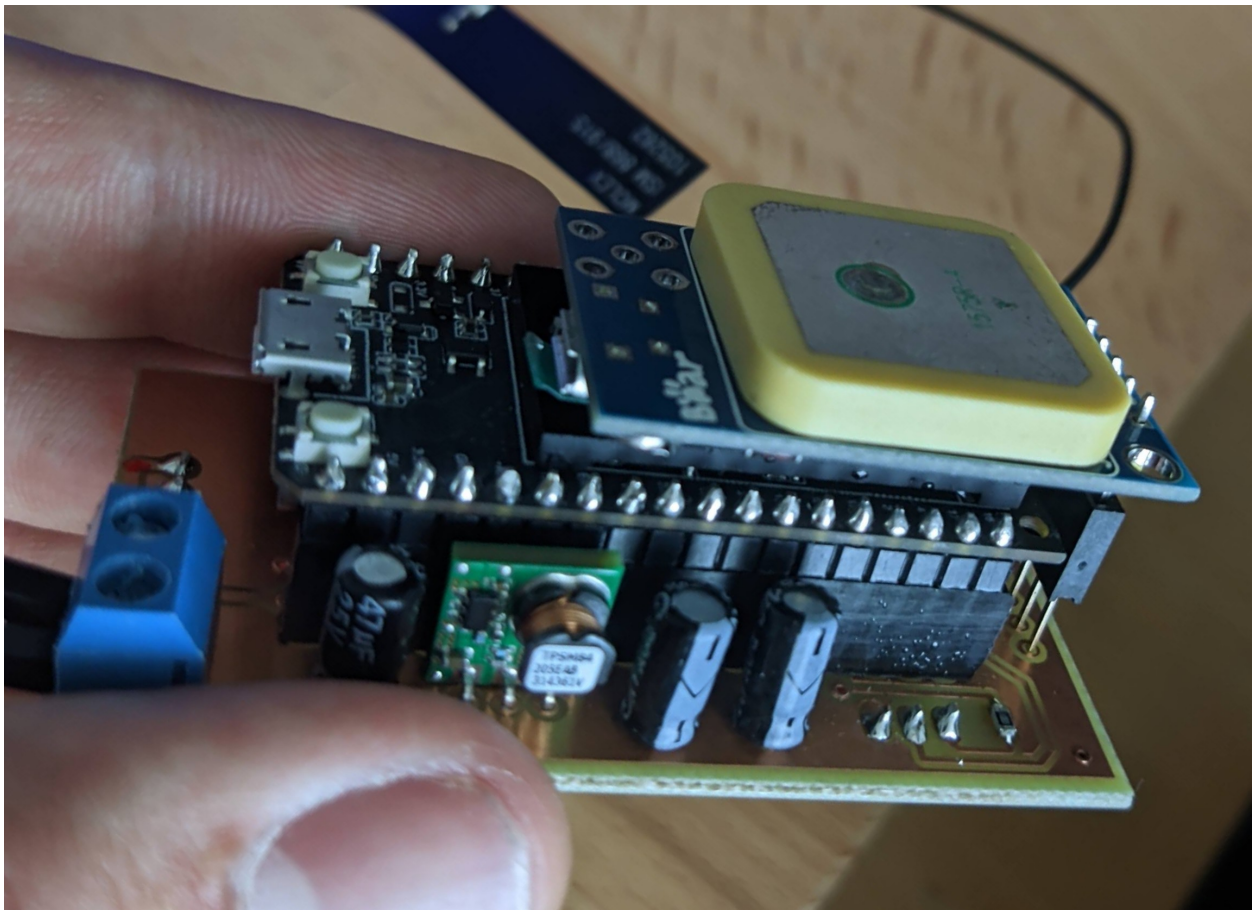


Figure 44: Final PCB with all the components mounted

6 Prototype evolution

The essence of prototyping is trying different approaches and versions to find the desired and more optimal result. This is achieved by trying and failing, but writing down the changes and problems of every version can help in the development. That is why in this section, the evolution between versions of the different parts of the add-on will be shown. By doing this, the thinking process behind every modification or improvement will be recorded, this way is easier to understand the progress and changes done.

The criteria to consider or discern between versions are the following. In the case of 3D designs, each print will be considered a milestone/version. Every batch of metal connectors will be written down. For the PCB same, each different physical version will be included in this section. Furthermore, a production budget will be also included.

6.1 Battery connector

6.1.1 Version 1

Created to check if it was possible to replicate the battery connector.

- No hollow space is considered for the hardware yet.
- Fits well.
- Supports included
- The base does not stick properly and as a result, it is bent.
- Springs in the tool push the piece out.
- Print can be seen in Figure 39

6.1.2 Version 2

- Included the battery grip to imitate the real shape
- Maximize the hollow space which stores the components
- Space created for the metal connectors to fit in between the walls and to reach the inside of the add-on so can be connected to the PCB
- Metal connectors from the power tool align perfectly with the holes
- Supports are needed to achieve a good result when printing the hollow space.

6.1.3 Version 3

- Created a new connector space for easier testing. It slides in and out
- Printed with a Prusa mk3s at *DTU Skylab* (see Figure 45)

Figure 45: Battery Connector final version with metal connectors

6.2 Power tool connector

6.2.1 Version 1

From a solid cube the existing and functioning 6.1.1 was subtracted. Modifications were needed to adjust to reality.

- Battery enters half way
- After some sanding and brute force, it fitted
- Errors were found as some parts of the structure were damaged by the brute force
- Base not perfectly straight
- Supports needed for the inside
- Print can be seen in Figure 40

6.2.2 Version 2

- Printing the piece in the Y-axis of the printer to make the piece base stick better to the bed
- Base is still not perfectly flat
- Base was designed to match 6.1.2 and it kind of did but flat bases are needed in both pieces to perfectly match
- Added hollow part on top to have more space for hardware (combined with the one in 6.1.2)
- Added space for the two small spikes that the battery has on the front
- Added a hollow place for the moving part of the battery that secures it in place
- Still not fitting perfectly and the secure moving part does not fall into place
- Some destruction was done to 6.2.1 to fully understand the design and improve the fitting

6.2.3 Version 3

- Battery secure flap that did not match, corrected.
- Possible impediments removed.
- Walls made 5mm taller so there is plenty of room inside
- Printed with a Prusa mk3s at *DTU Skylab* (see Figure 46)
- Consider creating a place to fit the two springs, so the tool's battery can be easily removed

Figure 46: Power tool Connector final version with metal connectors

6.3 Metal connectors

To make the installation of the add-on convenient, and quick. Keeping the slide-in functionality of the battery by reusing the existing connectors is a must. This way the add-on can be powered meanwhile allowing the normal behaviour of the tool. The connectors are proprietary and spare parts are not for sale, so different alternatives were searched.

The company Amphenol [67] offers customized battery or charger connectors and terminals meeting general industry standards. An email was sent to them explaining the desired connector but there was no response from the company.

Also, research was done in Digi-Key but none of the options was compatible. 3D conductive filament and conductive paint were also considered. However, the possibility of losing effectiveness with the wear and tear, and as it needed to bear a decent amount of current, hence these ideas were discarded.

Finally, the most appropriate idea seemed to be to handcraft them. For this, copper plates or other metallic pieces available at the lab will be used. The first approach was trying to mimic them by using the positive pole AA battery's metal connector plates. After a small trial batch (see Figure 47), they were discarded because of the poor functionality, low reliability and reduced versatility.

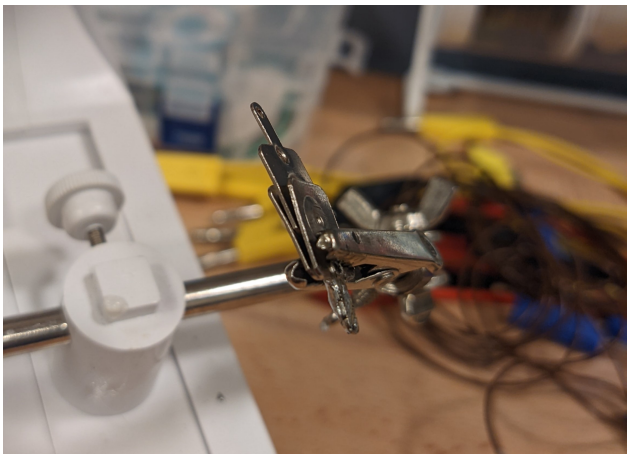


Figure 47: First test using positive plates for AA batteries

For a more suitable result, the final version where first designed in Fusion360 using the *sheet metal* menu, and then produced in three different metal plates to test the differences. The available plates were: 0.5 mm copper, 1mm copper and 1mm aluminium. For this task, a sheet metal nibbler, a metal file and pliers were used to cut, sand and bend the pieces (the tools used can be seen in Figure 48).

Figure 48: Hand tools used to create the metal connectors

In the following subsections, the production of both, the male and female connectors will be displayed and explained.

6.3.1 Male

For the male, a similar piece to the one in the tool was designed. For production, a stiff, straight 1mm thick copper piece was cut. Aluminium and 0.5mm copper were discarded because of stiffness and not being thick enough. Notice the cut on the top corner so it can fit tight in the corresponding part of the add-on. The long leg is for soldering the cable on the inside of the add-on. The design and implementation can be seen in Figure 49.

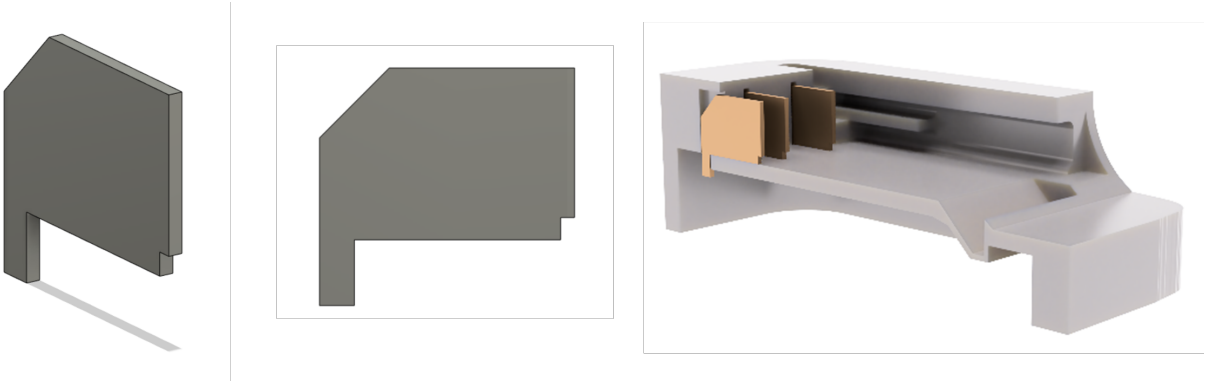


Figure 49: Male metal connector

6.3.2 Female

After opening the original battery and taking a look at the original connectors, a similar approach was taken. The connector is U-shaped but with the upper ends almost joined and slightly bent to the outside. Therefore they exert force against the walls and make the connection more robust. It also has a long leg facing down (for design requirements), where the cables will be soldered (see Figure 50). This makes them fit perfectly into the enclosure and stay in position while connecting and disconnecting. The chosen material was 0.5mm copper. The aluminium started to lose the closed shape after a few tests and the 1mm copper was too thick for such a small piece and really difficult to properly bend it.

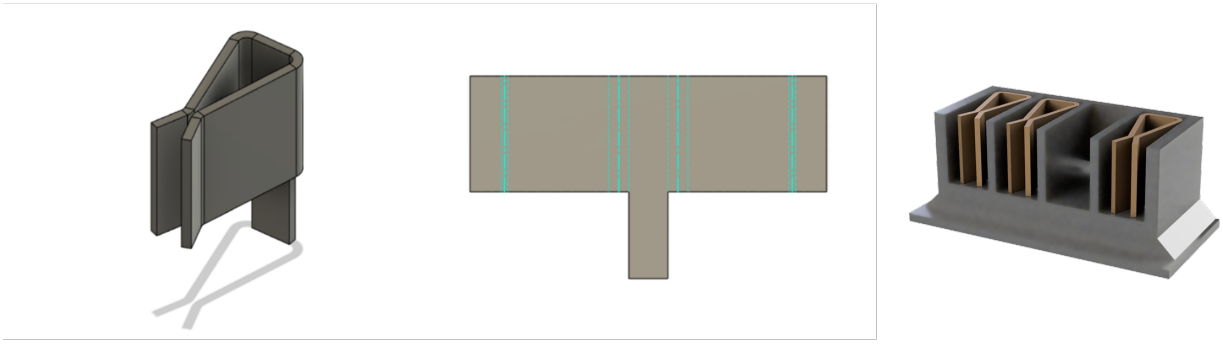


Figure 50: Female metal connector

6.4 PCB evolution

The first produced PCB had a weird error when it was printed and one of the ratsnet was not generated properly. It was only noticed after the etching and doing some holes. Unfortunately, it could not be reused. The following batch was checked closely and everything went fine. Two PCBs were etched at the same time. One was drilled and all the components soldered to it whereas the other one was kept as a backup.

6.5 Add-on prototype cost

As any product aimed to be sold. Price, printing time, production cost, cost of maintenance, PCB price, and components are to be considered.

For the maintenance, only three things must be considered. Replacement of the LiPo backup battery (every few years), pay Azure backedn and pay the LoRaWAN network provider. Now the free student version of Azure was used but the billing depends a lot on the usage. No exact price can be extracted from the used done as it has been irregular but must be considered in the future. Helium charges 1 Data Credit (DC) for every 24 bytes sent. Having in mind the possible uplink messages we can say that the most expected message is (3AP) or (2AP) thus 35 bytes on average, approximately 1.5 DC per transmission. Besides, the average sending interval is every 5 minutes, so it is a total of 288 messages per day.

One Data Credit equals $ 0.00001 USD. Therefore:

- Price per day $= 0.00001 * 1.5 * 288 = 0.00432\$$
- Price per month $= 0.00001 * 1.5 * 288 * 30 = 0.1296\$$
- Price per year $= 0.00001 * 1.5 * 288 * 365 = 1.5768\$$.

Disclaimer, this is a theoretical use case with constant use of the tracking device. Some business cases may not require every day sending messages (i.e. rental company when the tool is not being rented). The pay-per-use accounting of Helium brings more flexibility to companies and they do not have to worry about not taking full advantage of monthly subscriptions that other providers or technologies offer.

The Lilygo TTGO V1.3 micro controller can be bought from the manufacturer site for 21.88$ + shipping [68]

The Ublox Neo 7M GPS module can be bought for 215.00 DKK[**?**].

The 450 mAh battery could not be found online, but the price for the other 1800 mAh battery is 127.51 DKK. A slighty lower cost of 100 DKK will be used as an approximation.

PCB cost in JLCPCB for 5 boards is 73.92$ including shipping.

The DC-DC regulator costs 49.54 DKK [69] per unit.

Resistors, capacitors, pins, diodes, and connectors have an approximate cost of 4$. Price is difficult to estimate because depends on the number of components ordered.

In terms of 3D printing duration. For the final prototype, it was 15h and 15 minutes. If the pieces are ordered on the same website as the PCB, the price is 16.43$.

In total, the price to replicate the prototype and run it for one year including VAT is 111,99 $ or 765,72 *DKK*. This is without taking labour into account nor Azure. For mass production, the price will be lower.

Figure 51: Picture of the final assembled prototype

7 Microcontroller code

For programming the TTGo microcontroller, the Arduino IDE was used. As it is based on an ESP-32, when installing the corresponding library it is included as an available board.

First of all, the libraries for the modules are imported. In this case the *WIFI* (Wifi scanning), *SPI* (communicate with the LoRa module), *SoftwareSerial* (connect with the GNSS module), *TinyGPSPlus* (calculates the location from the *almanac* and *ephemeris* data). For *LoRaWAN*, the Arduino IBM C-library (*LMIC*) [70] was used. It is a portable implementation of the LoRaWAN 1.0.3 end device (LMIC stands for LoRaWAN MAC in C). It supports the *EU-868*, *US-915*, *AU-915*, *AS-923*, *KR-920* and IN-866 variants of the specification and it can handle *Class A* and *Class B* devices. The library takes care of all logical *MAC* states and timing constraints and drives the SEMTECH *SX1272* or *SX1276* radio. For our use case, it was implemented a class A device using the *EU-868* frequency and the *SX1276* chip.

The included example *helium-otaa* was used as a base to develop the whole code and get to know the library. First, all the required variables and pins are defined. Functions for each module and sending packages are coded accordingly to the results from Section 9. Finally, in the *event* function, all the logic is implemented according to the Diagram 24. This function is triggered when all the different *LoRaWAN* events occur. The Tx interval has to be established too.

For more details see Appendix F where is the implemented code. Inline comments can be found as well.

8 Backend

In this project, the backend is reached when the Helium network receives the messages from the add-on. Once the message is forwarded by the gateway, a deparsing function written in JavaScript (see Appendix B) is executed in their servers. The function translates the payload, a HEX string, into a JSON object called "decoded:" and adds it to the original message (example of a deparsed message in Appendix D).

Afterwards, the JSON is forwarded to the Azure implementation where is received by a *IoT Hub* resource. The helium packet flow can be visually seen in Figure 52.

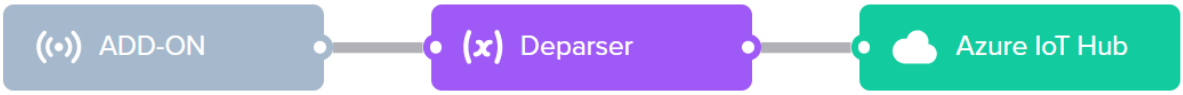


Figure 52: The message flow in Helium

To extract the information from the JSON object a *Stream Analytics job* is used. The *IoT Hub* resource is set as input. Every time a new message is received, the *Stream Analytics job* executes a Query code (see Appendix E) that extracts the useful information and sends it to *Power BI* where is displayed in a real-time dashboard (see Figure 53). *Power BI* also allows the creation of offline reports from the received data. With this functionality, a more pleasant presentation of the location can be accomplished as can be seen in Figure 54

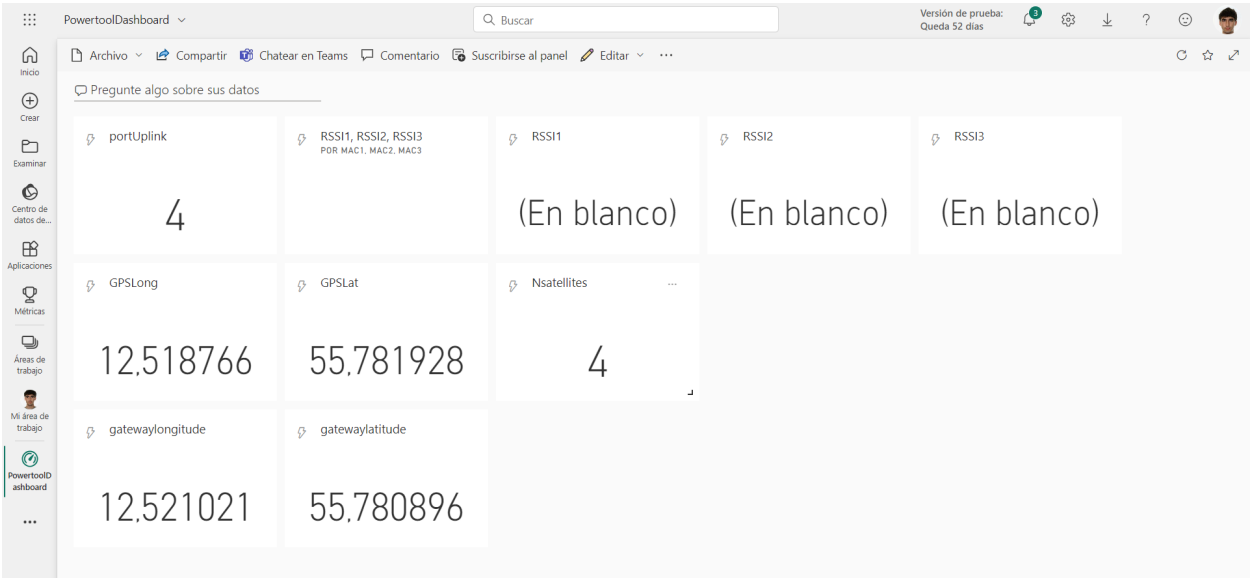


Figure 53: Dashboard in *Power BI* when a *GPS* package is received

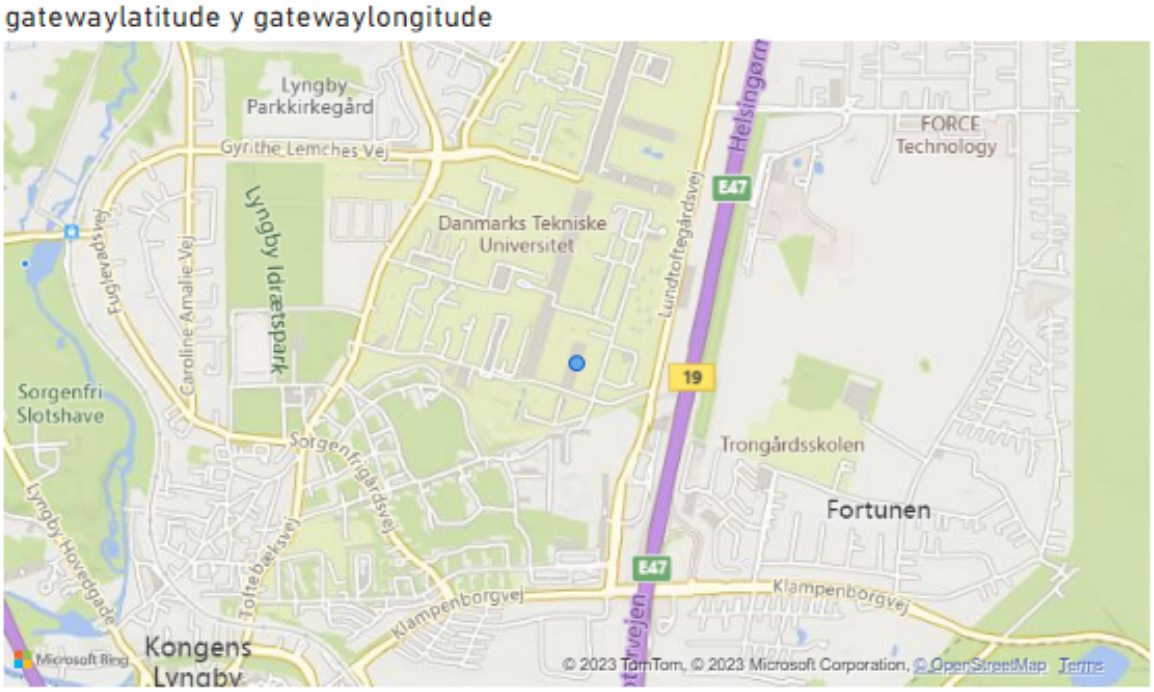


Figure 54: Power BI report map showing the gateway coordinates

9 Testing

In order to get to know our set-up, see the add-on consumption, the WiPS and GNSS capabilities and test all the functionalities, testing is a must. With this information, the code can be optimised and timed correctly according to the real performance of the modules. This way the expected result will be achieved and debugging can be saved.

9.1 Dimensions and weight

The add-on itself has 52.8 mm height, 106.1 mm length and 61 mm width. However, it only adds 48 mm of height to the power tool as it fits inside going from 213mm to 261mm tall.

The total add-on weight is 190 grams. Compared to the 1304g of the power tool, seems acceptable.

9.2 Power tool consumption

In order to be able to compare the consumption of the add-on with the consumption of the power tool measurements with different speeds were taken. The tool has a 10-clutch position, two drilling modes and 2 speeds. When pressing the trigger halfway, a light is turned on and when is fully pressed the motor starts. In Table **??** the current drained by some configurations were measured using a multimeter. The measurements were taken after a few seconds of triggering the tool and without load so that they were stable. Otherwise, when the motor starts to spin there is a spike or also under heavy load. Trying to measure the max current drained, a peak of 19.895 was recorded. This was achieved by carefully and manually stopping the tool because the max current the multimeter can handle is 20A. The real max peak must be higher than that measure but not much more because the tool automatically stops at some point to prevent damage.

Mode	**Current**
Just light	0.032 A
Speed 1	1.65A
Speed 2	2.6 A

Table 4: Current drained by the power tool

9.3 Add-on power consumption

As we are adding more load to the battery is interesting to see how much our device drains. This way, we can know how long will it last on standby, see if it is power efficient and consider changes to reduce this power consumption. Instead of using a multimeter to measure the current of the add-on, an oscilloscope will be used to achieve more accuracy. Oscilloscopes are meant to measure voltage, so we need to translate current into voltage. To

do this, a shunt resistor will be set as a load and the oscilloscope will measure the voltage drop on it. By knowing the exact value of the resistor and how much it dropped, the current can be calculated using Ohm's law $I = \frac{V_{measured}}{R}$. For easier calculations, 1Ω resistors or other values multiples of 10 are used. The watts that this resistor has to dissipate need to be taken into account. Being especially careful with relatively high currents.

Let's do the calculations for the worst-case scenario of the tool, when it drains 20A, using the formula $P = RI^2$. If the resistor is 1Ω $P = 1*20^2 = 400W$ it will require a substantial resistor able to handle that much power. However, if the resistor is 0.01Ω $P = 0.01*20^2 = 4W$ which is a more bearable value and the resistor will not need to be that big. Nevertheless, most of the time a piece of constantan cable is used. The manufacturer provides the electrical resistivity (ρ) in Ωm^2, this way the cable can be cut accordingly to the desired resistor value. This material's main feature is the low thermal variation of its resistivity.

The relation between the battery voltage drop and the increase in current drained by the tool can be seen using the oscilloscope. By connecting one of the channels to the tool's battery and the other channel to the resistor and turning on the tool. This way the internal battery resistance can be measured [71]. Unluckily, no constantan nor big shunt resistors were available in the lab, so the current of the tool can not be measured with the oscilloscope.

In our case, the *Hantek DSO5202P* oscilloscope has a resolution of 1 mW. With a 1Ω resistor it will be enough to measure the add-on consumption as the expected voltage drop is of the order of tens of mV. The peak-to-peak voltage detected was $5.12mV$ 55 meaning that the current drained was about 5.12 mAh a really low value for the add-on consumption considering the *ESP32-D0WDQ6* chip consumes at least 20 mA 31 mA while awake [49]. This can be because of the noticeable ripple of the DC-DC regulator or the precision loss for measuring close to the limit.

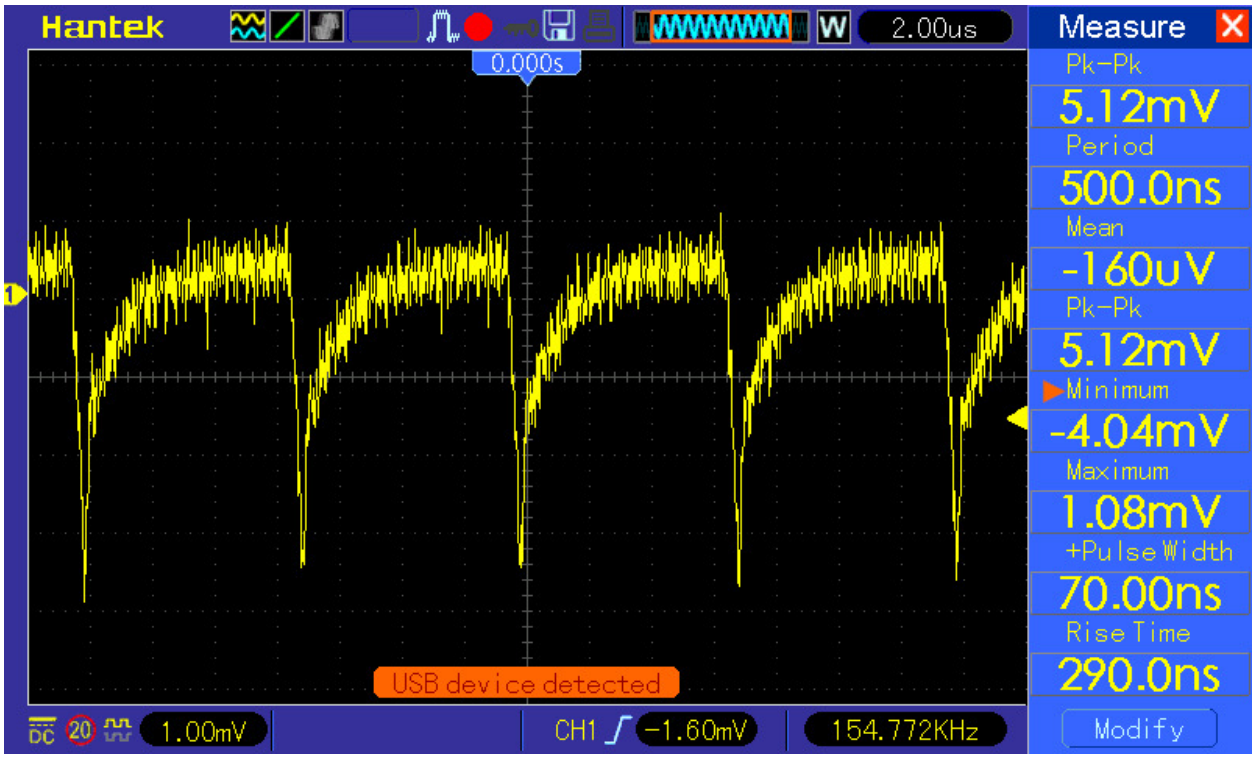


Figure 55: Oscilloscope screenshot while measuring the voltage drop in a 1Ω resistor

The industrial digital multimeter *UT171A* was used instead to get precise current measurements. The add-on was tested in the different possible situations and the Table **??** was built using the obtained values.

SITUATIONS	CURRENT
Microcontroller disconected	0.174 mA
Sleeping	1.719 mA
Standby	13.510 mA
Standby charging battery at 75%)	117.3 mA
WiFi scanning	35.471 mA
GNSS module ON	39.184 mA
LMIC standby	43.1 mA
Sending through LoRa	55.2 mA

Table 5: Add-on current drained in different situations

Taking into account that the tool consumption in Table **??**, the values in Table **??** seem low.

9.4 Heat dissipation

Due to the low consumption values obtained in Section 9.3 no dissipation calculations were done. Furthermore, this is the first prototype and the hardware may change in the future. However, this test will be necessary for the final project as it is an enclosed add-on without ventilation. Checking with the operating temperature ranges that the different components support.

9.5 GNSS module real measurements

In order to truly know how much time the implemented module takes to fix satellites considering the attenuation inside the add-on, real-world tests were carried out. The module was placed in its final position where it was initialized using the U-center [72] app and timed until a location was fixed. Both kinds, hot and cold starts were tested in six different potential situations, half indoors and the other half outdoors (see Table **??** to see the results).

	Test	**Cold start**	**Hot start**
Indoors	1 (next to a window)	260 s	27 s
	2 (living room)	276 s	169s
	3 (corridor)	No fix	No fix
Outdoors	1 (clear sky view)	35 s	2 s
	2 (next to a building)	34 s	2 s
	3 (leafy forest)	283 s	68 s

We have to be very careful comparing the times given by the manufacturer in Table **??** with the obtained in the real testing Table **??** as not the same criteria were followed. The chip manufacturer stopped the timer when the chip fixed the satellites with a gain of -130 *dbm* and in the real-world module test, it was timed until a location was retrieved. Besides, the manufacturer most likely tested the Neo 7-N chip under ideal circumstances and with a professional antenna.

This highlights the relevance of doing real-world tests. This way you get to know your own set-up, the capabilities of the module and your real-case scenarios. Furthermore, this information is useful when coding the microcontroller to optimize power consumption and to know the mean expected time to fix.

During the testing, a concrete behaviour was found. The module tends to fix mainly *GPS* satellites even though it supports other engines. Very punctual *GLONASS* or *SBAS* satellites were also fixed, but rarely or not for long. The possibilities offered in the configuration menu were analysed trying to make the module use all of them and hopefully improving the fixing time. The changes could not be implemented through the *U-center* but maybe sending commands using the *UART* interface could be successful (not tried). It can also be something hard-coded by the manufacturer of the final module but no information was

found about it.

9.6 Skyhook performance test

With the help of an *ESP-32*, real field tests were carried out to see the precision of the *API*. Trying to find the best balance between accuracy and data size due to *LoRaWAN* duty-cycle restrictions and Helium billing. In Table **??**, you can see the three tests done in different locations. For each test, a scan was captured and stored the *MAC* and *RSSI* of each access point. Finally, the *Postman* API platform [73] is used to *POST* to *Skyhook* and get the different accuracy values.

	Test 1 (DTU library)	**Test 2 (Nærum Gymnasium)**	**Test 3 (Nærum Hovedgade)**
1 AP	10 (4 decimals)	10 (4 decimals)	10 (4 decimals)
2 APs	10 (4 decimals)	33	25
3 APs	34	30	13
ALL APs	30 (13 APs)	22 (6APs)	13 (23 APs)

Table 6: Skyhook accuracy values in different environments and with different amounts of APs

Disclaimer, no documentation talking about the accuracy value by *SkyHook* was found. It was assumed from the testing that the lower the better. Some potential *WiFi* features such as channel or *SSID* are omitted because they do not contribute to the precision and are redundant. When doing the scan, the different *APs* were sorted per *RSSI* and selected the higher components. Also, the tests that took part in DTU were a bit challenging because the same AP hosts different networks, so different *SSID* and *MACs* point to the same physical router. Because of this, the accuracy is not that great. An algorithm could reduce these effects and filter frankly similar or consecutive *MAC* addresses (i.e. see Appendix C) and try to avoid mobile hotspots.

From this testing, we can tell that 3 different access points are the most optimal amount to achieve good precision while not sending too much information. Also when only using one AP the precision is very low providing only 4 decimals to the coordinates, hence it must be avoided.

10 Conclusion

After a long journey, the main goal of creating a working prototype was achieved. A functional add-on enclosure was designed from a 3D scan of the tool and battery. The add-on is split into two parts to have easy access to the inside space. After 3 versions, the final 3D-printed pieces were obtained. No joining solution for the pieces was developed. Also, the metal connectors integrate well, are functional and allow the normal behaviour of the tool. Electronics are combined in the custom hand-made *PCB*. It took a long time to properly design the PCB and produce it but the result is very satisfying. For the communication, *LoRaWAN* was implemented but the coverage in reality is not that great. So maybe a different *LPWAN* technology should be considered instead to avoid no coverage areas. The monitoring functionality is a simple battery level value but no more was expected as the add-on has only access to the battery.

The backend is a very broad element. After difficulties using *Azure*, a live dashboard of the values in *Power BI* was accomplished. The *API* call to *SkyHook* was not automated, nor were the downlink configuration messages. To test these functionalities, *Postman API Platform* and the *Helium Console* were used. *Azure* is a very powerful tool that allows many applications and scalability but for the first prototype, it was too complex.

It was a big challenge to adjust the scope of the project and match the expected working load. Lots of ideas were considered but could not be included because of time. In the design and implementation process of this system, many different areas were involved. A theoretical overview of them was carried out to get into context. All of them are crucial to obtain a working prototype. Therefore, a minimum development in each area had to be achieved, leaving little time to exhaustively work on all the parts.

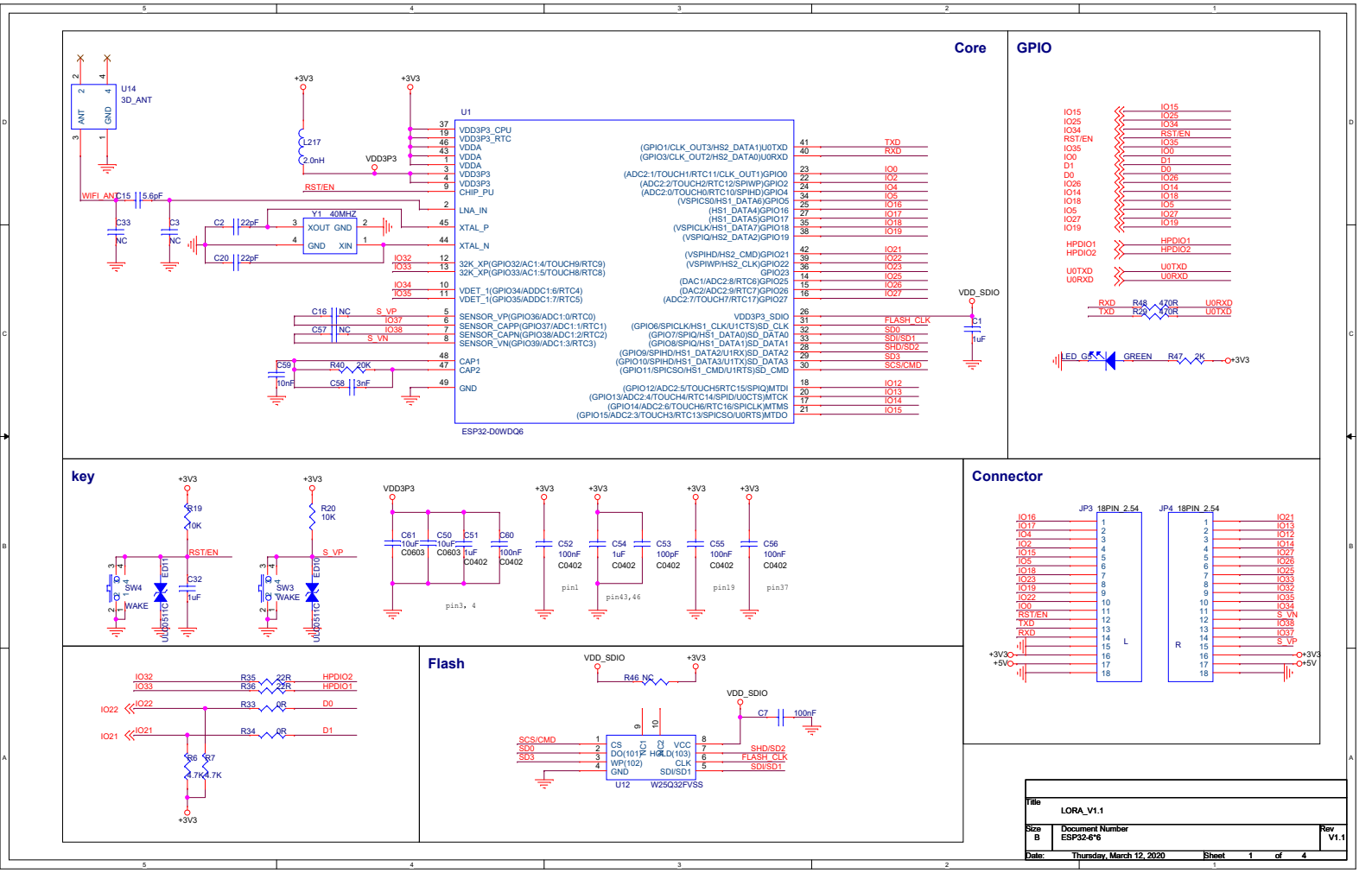
11 Future Work

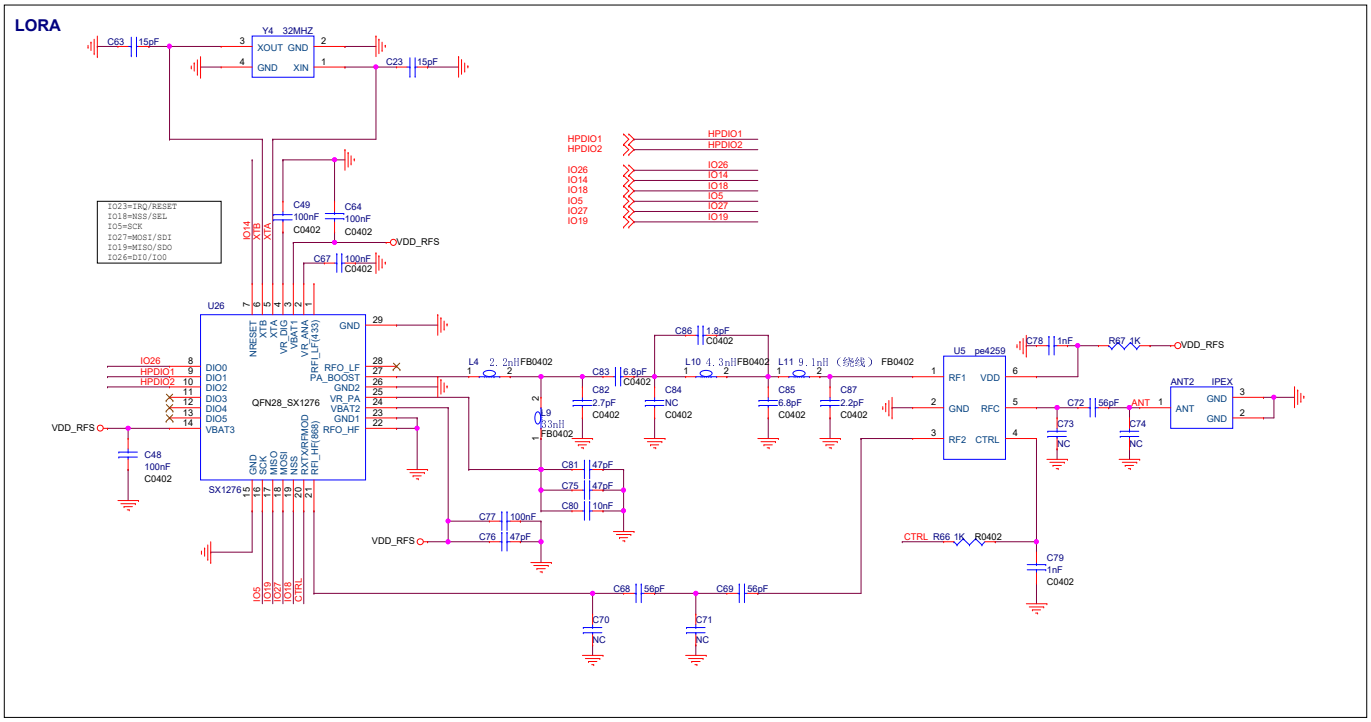
During the development of the add-on, many new ideas or things to improve were found. As they are out of the scope for this first prototype or more oriented towards the final project, they were collected in this list:

- Include BLE beacons to precisely find lost tools and to easily configure the device
- Improve efficiency (reduce messages during the night, do not send messages if the information is similar, deep-sleep mode)
- Further testing (minimum sending time between messages, detailed power budget, test run of several days ...)
- Log info and do more analysis of data
- Over-discharge protection for the tool's battery
- Create different predefined modes of behaviour (lost item, non-used item, high-risk item, low-risk item ...)
- Consider different ways of coding the information to minimize transmission errors and use Huffman codification to reduce package size
- Use chips instead of modules to produce the PCB
- Smart algorithm to select the best APs to send to the backend (*SkyHook* accuracy problem)
- Proper way to join both parts of the add-on
- Implement an interruption in the microcontroller to detect falls in the voltage divider and send a message when the main battery is disconnected.
- Having to change *LoRaWAN* frequency depending on the country.
- Use the SLA printer for more accurate and stronger results
- Incorporate springs in the add-on for quick release
- Fully integrate the hardware inside the tool
- Compare Skyhook with other alternatives such as Wigle and Google API
- Add another mosfet to cut the signal from the 3rd cable when wanting to stop the tool

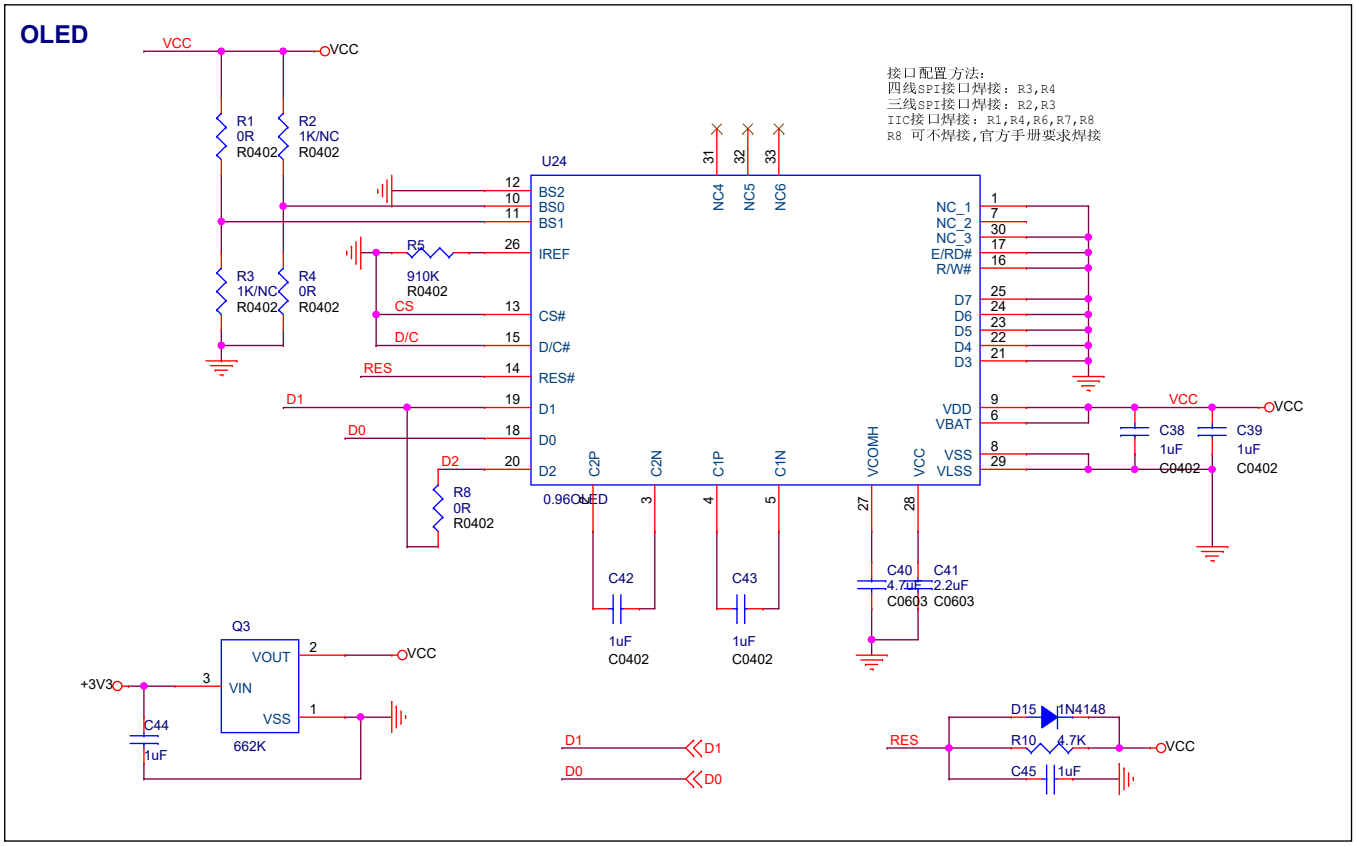
Appendices

A LILYGO TTGO LORA V1.3 schematic

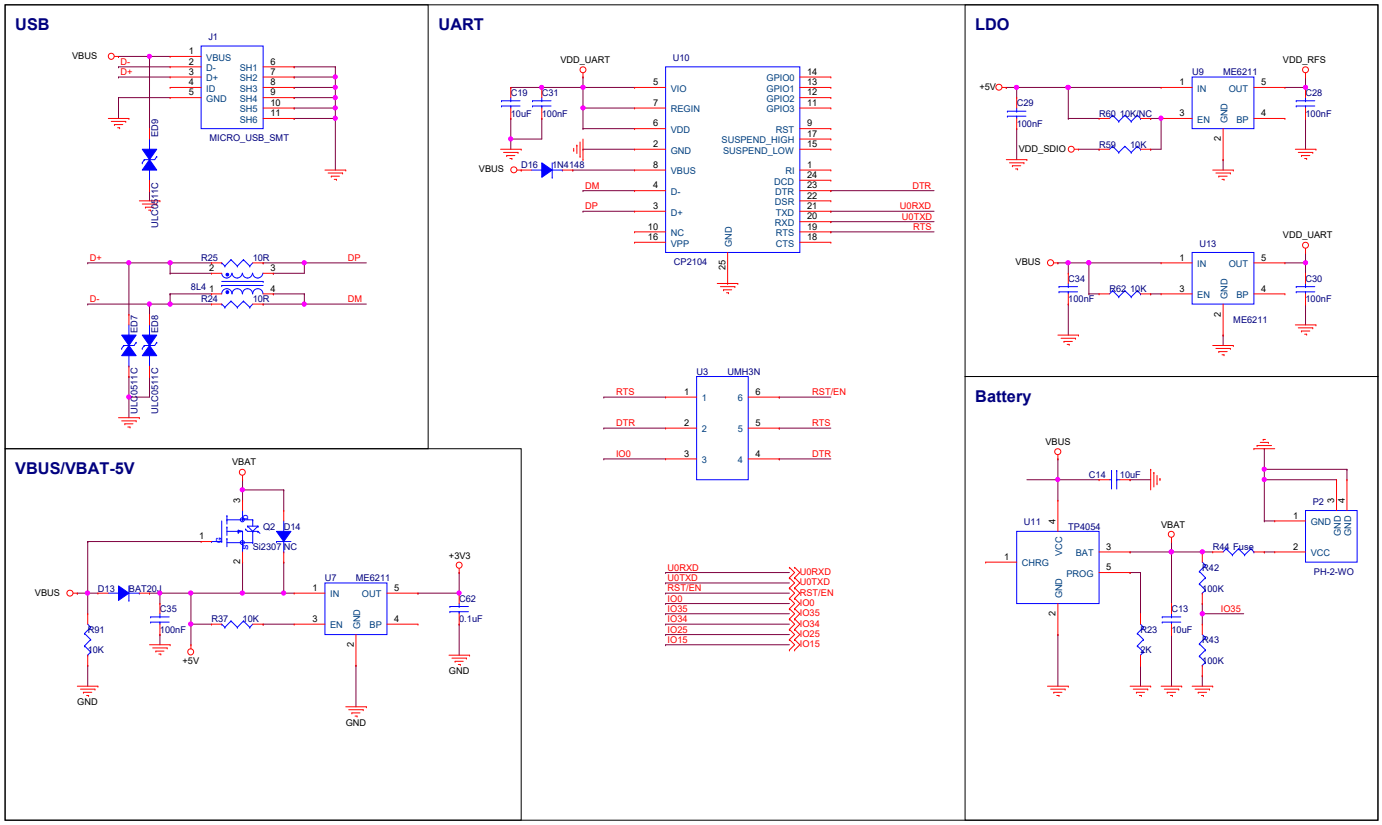

Core
GPIO
key
Connector
Flash
U14
3D_ANT
ANT
GND
WIFI_ANT
L217
2.0nH
VDD3P3
RST/EN
U1
ESP32-D0WDQ6
VDD3P3_CPU
VDD3P3_RTC
VDDA
LNA_IN
XTAL_P
XTAL_N
CHIP_PU
Y1 40MHZ
XOUT GND
GND XIN
C2 22pF
C20 22pF
C33 NC
C3 NC
C15 5.6pF
32K_XP(GPIO32/AC1:4/TOUCH9/RTC9)
32K_XP(GPIO33/AC1:5/TOUCH8/RTC8)
VDET_1(GPIO34/ADDC1:6/RTC4)
VDET_1(GPIO35/ADDC1:7/RTC5)
SENSOR_VP(GPIO36/ADC1:0/RTC0)
SENSOR_CAPP(GPIO37/ADC1:1/RTC1)
SENSOR_CAPN(GPIO38/ADC1:2/RTC2)
SENSOR_VN(GPIO39/ADC1:3/RTC3)
CAP1
CAP2
GND
VDD3P3_SDIO
VDD_SDIO
C1 1uF
C16 NC
C57 NC
C59 10nF
R40 20K
C58 3nF
S_VP
TXD
RXD
FLASH_CLK
SD0
SDI/SD1
SHD/SD2
SD3
SCS/CMD
IO0
IO2
IO4
IO5
IO12
IO13
IO14
IO15
IO16
IO18
IO19
IO21
IO22
IO23
IO25
IO26
IO27
IO32
IO33
IO34
IO35
IO37
IO38
HPDIO1
HPDIO2
U0TXD
U0RXD
RST/EN
RXD
TXD
R48 470R
R29 470R
LED_GREEN
R47 2K
+3V3
+5V
R19 10K
R20 10K
SW4 WAKE
SW3 WAKE
C32 1uF
ED11
ED10
U0603TC
C61 10uF C0603
C50 10uF C0603
C51 1uF C0402
C60 100nF C0402
C52 100nF C0402
C54 1uF C0402
C53 100pF C0402
C55 100nF C0402
C56 100nF C0402
pin3, 4
pin1
pin43,46
pin19
pin37
JP3 18PIN 2.54
JP4 18PIN 2.54
L
R
R35 22R
R36 22R
R33 0R
R34 0R
R6 4.7K
R7 4.7K
HPDIO2
HPDIO1
D0
D1
R46 NC
C7 100nF
U12
W25Q32FVSS
CS
DO(IO1)
WP(IO2)
GND
VCC
HOLD(IO3)
CLK
SDI/SD1
Title
LORA_V1.1
Size B
Document Number ESP32-6*6
Rev V1.1
Date: Thursday, March 12, 2020
Sheet 1 of 4



Title	LORA_V1.3	
Size B	Document Number TTGO LORA	Rev V1.3
Date: Thursday, March 12, 2020	Sheet 2 of 4	



Title	LORA_V1.3		
Size A	Document Number OLED		Rev V1.3
Date: Thursday, March 12, 2020		Sheet 3 of 4	



Title	LORA_V1.3	
Size B	Document Number POWER	Rev V1.3
Date: Thursday, March 12, 2020	Sheet 4 of 4	

B WiFi scan at the DTU library

SSID	RSSI	CH	MAC address
eduroam	-80	11	28:6F:7F:78:F0:A0
DTUsecure	-80	11	28:6F:7F:78:F0:A1
device	-80	11	28:6F:7F:78:F0:A2
DTUdevice	-81	11	28:6F:7F:78:F0:A3
DTUguest	-81	11	28:6F:7F:78:F0:A4
DTUsecure	-83	6	28:6F:7F:3A:2F:21
DTUguest	-83	6	28:6F:7F:3A:2F:24
eduroam	-84	6	28:6F:7F:3A:2F:20
DTUdevice	-84	6	28:6F:7F:3A:2F:23
device	-85	6	28:6F:7F:3A:2F:22
DTUdevice	-85	1	A0:E0:AF:71:3B:A2
AI-THINKER_0BA95	-89	6	2E:3A:E8:0B:A9:54
iPhone	-89	6	F6:4A:17:7C:2B:B2

C Java Script Deparser of the payload

```
function Decoder(bytesArray, port) {

  var obj = new Object();

  switch(port) {
  case 1: // 1 AP
    obj.mac1 = bytesArray.slice(0, 6).map(byte => byte.toString(16).padSta
    obj.rssi1 = "-" +bytesArray[6].toString(16);
    obj.battery = bytesArray[7].toString(16);
    obj.Nsatellites = 0;
    obj.lat = 0;
    obj.lng = 0;
    break;

  case 2: // 2 APs
    obj.mac1 = bytesArray.slice(0, 6).map(byte => byte.toString(16).padSta
    obj.rssi1 = "-" +bytesArray[6].toString(16);
    obj.mac2 = bytesArray.slice(7, 13).map(byte => byte.toString(16).padSt
```

```
    obj.rssi2 ="-" + bytesArray[13].toString(16);
    obj.battery = bytesArray[14].toString(16);
    obj.Nsatellites = 0;
    obj.lat = 0;
    obj.lng = 0;
    break;

  case 3: // 3 APs
    obj.mac1 = bytesArray.slice(0, 6).map(byte => byte.toString(16).padSta
    obj.rssi1 ="-" + bytesArray[6].toString(16);
    obj.mac2 = bytesArray.slice(7, 13).map(byte => byte.toString(16).padSt
    obj.rssi2 ="-" + bytesArray[13].toString(16);
    obj.mac3 = bytesArray.slice(14, 20).map(byte => byte.toString(16).padS
    obj.rssi3 = "-" +bytesArray[20].toString(16);
    obj.battery = bytesArray[21].toString(16);
    obj.Nsatellites = 0;
    obj.lat = 0;
    obj.lng = 0;
    break;

  case 4: //GPS
    const ascii = decimalArrayToAscii(bytesArray);
    //obj.ascii = ascii;
    obj.Nsatellites = ascii.slice(0, 1);
    obj.lat = ascii.slice(1, 10);
    obj.lng = ascii.slice(10, 20);
    obj.battery = bytesArray[20].toString(16);

  break;
}

  return obj;
}


function decimalArrayToAscii(decimalArray) {
  let asciiString = '';

  for (let i = 0; i < decimalArray.length; i++) {
    const decimalValue = decimalArray[i];
    const asciiCharacter = String.fromCharCode(decimalValue);
    asciiString += asciiCharacter;
  }
```

```
  return asciiString;
}
```

D JSON deparsed message


```
{    "app_eui": "6081F9E6D8338XXX",
     "dev_eui": "6081F937B01EEXXX",
     "devaddr": "E2000XXX",

"_unmodeleddata": {
     "dc": {
          "balance": 100,
          "nonce": 1
     },
     "decoded": {
          "payload": {
               "Nsatellites": 0,
               "lat": 0,
               "lng": 0,
               "mac1": "30:30:32:41:31:30",
               "mac2": "34:41:37:39:30:37",
               "mac3": "30:30:32:41:31:30",
               "rssi1": "-46",
               "rssi2": "-32",
               "rssi3": "-46"
          },
          "status": "success"
     },
     "fcnt": 1,
     "hotspots": [{
          "channel": 5,
          "frequency": 868.1,
          "hold_time": 0,
          "id": "112Q4RnQY3zhaMbqRhBerVRAnPkEtviU9GbTPpqSxiXBNFkw",
          "lat": 55.78089643719182,
          "long": 12.521021432894424,
          "name": "stale-tangelo-giraffe",
          "reported_at": 1687433012364,
          "rssi": -109,
```

```
            "snr": 6.5,
            "spreading": "SF7BW125",
            "status": "success"
        }],
        "id": "a58d755a-7cf5-4e42-9f5c-70513d62b",
        "metadata": {
            "adr_allowed": false,
            "cf_list_enabled": false,
            "multi_buy": 1,
            "organization_id": "3c057ee-4399-a76e-7fb4f376b1",
            "preferred_hotspots": [],
            "rx_delay": 1,
            "rx_delay_actual": 1,
            "rx_delay_state": "rx_delay_established"
        },
        "name": "TTGO",
        "payload": "MDAyQTEwRjRBNzkwNzIwMDJBMTBGNEE3OTE3MjAwMkExMEY0QTc5Mjcy",
        "payload_size": 42,
        "port": 3,
        "raw_packet": "QOIAAEiAAQADABPRvWWI2sTQm6bzpr9FFeRFCu29WT+Op9yAMtaIghh
        "replay": false,
        "reported_at": 1687433012364,
        "type": "uplink",
        "uuid": "b2c919-c2e8-4f21-95aa-10f2015b"
    },
    "_eventtype": "Telemetry",
    "_timestamp": "2023-06-22T11:23:34.439Z"
    }
```


E Query code

```
SELECT
    CAST(iothub.EnqueuedTime AS datetime) AS event_date,
    CAST(GetArrayElement(inputhubowner.hotspots, 0).lat AS float) AS gatew
    CAST(GetArrayElement(inputhubowner.hotspots, 0).long AS float) AS gate
    CAST(payload AS nvarchar(max)) AS hexpayload,
    CAST(port AS bigint) AS portUplink,
    CAST(decoded.payload.mac1 AS nvarchar(max)) AS MAC1,
    CAST(decoded.payload.mac2 AS nvarchar(max)) AS MAC2,
    CAST(decoded.payload.mac3 AS nvarchar(max)) AS MAC3,
    CAST(decoded.payload.rssi1 AS bigint) AS RSSI1,
    CAST(decoded.payload.rssi2 AS bigint) AS RSSI2,
```

```
    CAST(decoded.payload.rssi3 AS bigint) AS RSSI3,
    CAST(decoded.payload.Nsatellites  AS bigint) AS Nsatellites,
    CAST(decoded.payload.lat  AS bigint) AS GPSLat,
    CAST(decoded.payload.lng  AS bigint) AS GPSLong


INTO
    outputbi
FROM
    inputhubowner
```

F Microcontroller Arduino code

```
#include <WiFi.h>
#include <SPI.h>              // Comunicarion with the LoRa module
#include <lmic.h>             // LoRaWAN
#include <hal/hal.h>          // LoRaWAN
#include <TinyGPSPlus.h>      // GNSS data
#include <SoftwareSerial.h>   // For creating Serial comunications
    with regular digital pins


#define BUZZZER_PIN 0

const int vdPin = 34;    // Voltage divider Pin (ADC1_6)
int vdValue = 0;         // Variable to store the ADC value
int BatPercentage = 0;   // Variable to store the ADC value
int n_WIFI = 0;          // Variable to store the amount of WIFI
   AP's



bool GPS_flag = false;      // Flag for the GPS
bool Buzzer_flag = false;   // Flag for the buzzer


//////    GPS    ////////////////////////////////////////////
static const int RXPin = 17, TXPin = 16, GPS_mosfet = 2;  // Pins
```

```
25 static const uint32_t GPSBaud = 9600;                    //
     Baudrate
26
27 // Variables
28 int n_satellites;
29 double lat;
30 double lng;
31
32 TinyGPSPlus gps;                    // Create the GPS object
33 SoftwareSerial ss(RXPin, TXPin);  // Create the GPS serial
     communication
34
35 // Starting the UART communication with the TinyGPS object
36
37
38 void GPS_values() {
39   /*
40     GPS_values() reads the values from the module and if they are
         valid
41       and more precise (nÂş satellites >= previous nÂş satellites
           ) stores Latitude,
42       Longitude and NÂş of satellites
43
44     : no return but updates global variables
45   */
46
47   if ((gps.satellites.isValid() and gps.location.isValid()) and (
       gps.satellites.value() >= n_satellites)) {
48     Serial.println("GPS_values()_in_the_IF");
49     n_satellites = gps.satellites.value();
50     lat = gps.location.lat();
51     lng = gps.location.lng();
52     Serial.println(n_satellites);
53     Serial.println(lat);
54     Serial.println(lng);
55     Serial.println((n_satellites > 0) and ((lat != 0) and (lng !=
         0)));
56   }
57 }
58
59
60 static void smartDelayGPS(unsigned long ms) {
61   /*
62     smartDelayGPS(ms) does a delay while reading the values of
```

```
        the GPS
63        ms : amount of ms to apply the delay
64
65        : no return but updates global variables
66    */
67    Serial.println("smartDelayGPS");
68    Serial.println(ss.available());
69
70    unsigned long start = millis();
71    do {
72      if (ss.available() > 0) {
73        gps.encode(ss.read());
74        GPS_values();
75      }
76    } while (millis() - start < ms);
77 }
78 /////////////////  Message arrays
     ////////////////////////////////
79
80
81
82 char dataStringUplink[43] = { 0 };    // Max sending is 3 MAC's
     (3*12) + 3 SSID's (3*2) + Baterry % (1)
83 char dataStringDownlink[2] = { 0 };   // Receiving message 1 byte
84
85
86 void batteryLevel() {
87    /*
88      Gets the battery level by doing a conversion from ADC values
          to battery percentage.
89      The ADC value (3626) was obtained when the battery was fully
          charged.
90
91      : no return but updates global variables
92    */
93    vdValue = analogRead(vdPin);
94    BatPercentage = (vdValue * 100) / 3626;
95 }
96
97
98 ////////////   LoRaWAN
     ////////////////////////////////////////
99
100 // This EUI must be in little-endian format, so least-significant
```

```
    -byte first.
101 static const u1_t PROGMEM APPEUI[8] = { 0x1B, 0x88, 0x33, 0xD8, 0
    xE6, 0xF9, 0x81, 0x60 };
102 void os_getArtEui(u1_t* buf) {
103   memcpy_P(buf, APPEUI, 8);
104 }
105
106 // This should also be in little endian format, see above.
107 static const u1_t PROGMEM DEVEUI[8] = { 0xEE, 0xF1, 0x5A, 0xB0, 0
    x37, 0xF9, 0x81, 0x60 };
108 void os_getDevEui(u1_t* buf) {
109   memcpy_P(buf, DEVEUI, 8);
110 }
111
112 // This key should be in big endian format (or, since it is not
    really a
113 // number but a block of memory, endianness does not really apply
    ). In
114 // practice, a key taken from the Helium console can be copied as
    -is.
115 static const u1_t PROGMEM APPKEY[16] = { 0x13, 0x3F, 0xC5, 0x66,
    0x34, 0x9E, 0x4B, 0xE9, 0xCF, 0x8F, 0xE6, 0x35, 0x7A, 0x70, 0
    x22, 0x1E };
116 void os_getDevKey(u1_t* buf) {
117   memcpy_P(buf, APPKEY, 16);
118 }
119
120 static osjob_t sendjob;
121
122 // Schedule TX every this many seconds (might become longer due
    to duty
123 // cycle limitations).
124 const unsigned TX_INTERVAL = 300;  // Every 5 minutes
125
126 // Pin mapping for LORA transceiver
127 const lmic_pinmap lmic_pins = {
128   .nss = 18,
129   .rxtx = LMIC_UNUSED_PIN,
130   .rst = 23,
131   .dio = { 26, 33, 32 },
132 };
133
134 void CheckDownlinkMessage() {
135   /*
```

```
136   CheckDownlinkMessage() parses the received message. If it is an
          ACK print it. If it is a regular
137   message, the function parses it and decodes it from base 64 and
          adapt the flags to what the package said (following the
         custom protocol)
138
139     : no return, Serial.print and update of global variable
140   */
141
142   if (LMIC.txrxFlags & TXRX_ACK) {  // ACK message
143     Serial.println(F("Received␣ack"));
144   }
145
146   if (LMIC.dataLen != 0) {  // Downlink message
147     Serial.println(F("Received␣"));
148     int inputStringLength = sizeof(LMIC.dataLen);  // or strlen()
149     Serial.println(inputStringLength);
150     Serial.println(F("rawpayload"));
151
152     int value = int(LMIC.frame[0]);
153
154     switch (value) {  // Check the case and update the flags
155       case 0:
156         GPS_flag = false;
157         Buzzer_flag = false;
158         break;
159
160       case 1:
161         GPS_flag = true;
162         Buzzer_flag = false;
163         break;
164
165       case 2:
166         GPS_flag = false;
167         Buzzer_flag = true;
168         break;
169
170       case 3:
171         GPS_flag = true;
172         Buzzer_flag = true;
173         break;
174     }
175   }
176 }
```

```
177
178 void printHex2(unsigned v) {
179   /*
180   printHex2(unsigned v) translates Hex character to String and
        prints it on the Serial
181     v: HEX char
182
183     : no return, only Serial.print
184   */
185   v &= 0xff;
186   if (v < 16)
187     Serial.print('0');
188   Serial.print(v, HEX);
189 }
190
191
192 void BuzzerSound(int n) {
193   /*
194   BuzzerSound(int n) makes the buzzer sound for 1.5 s and turns
        it off for 1 s.
195   This is repeted as many times as indicated by the given
        parameter.
196
197   n: number of iterations (in this case beeps)
198   */
199   for (int i = 0; i < n; i++) {
200     tone(BUZZZER_PIN, 3000, 1500);
201     delay(1500);  // Delay is not very critic, no RX possible
202     noTone(BUZZZER_PIN);
203     delay(1000);
204   }
205 }
206
207
208 void onEvent(ev_t ev) {
209   /*
210   onEvent(ev_t ev) prints the events happening with the LoRaWAN
        connection
211
212   ev:
213   */
214   Serial.print(os_getTime());
215   Serial.print(": ");
216   switch (ev) {
```

```
217     case EV_SCAN_TIMEOUT:
218       Serial.println(F("EV_SCAN_TIMEOUT"));
219       break;
220     case EV_BEACON_FOUND:
221       Serial.println(F("EV_BEACON_FOUND"));
222       break;
223     case EV_BEACON_MISSED:
224       Serial.println(F("EV_BEACON_MISSED"));
225       break;
226     case EV_BEACON_TRACKED:
227       Serial.println(F("EV_BEACON_TRACKED"));
228       break;
229     case EV_JOINING:
230       Serial.println(F("EV_JOINING"));
231       break;
232     case EV_JOINED:
233       Serial.println(F("EV_JOINED"));
234       {
235         u4_t netid = 0;
236         devaddr_t devaddr = 0;
237         u1_t nwkKey[16];
238         u1_t artKey[16];
239         LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);
240         Serial.print("netid:␣");
241         Serial.println(netid, DEC);
242         Serial.print("devaddr:␣");
243         Serial.println(devaddr, HEX);
244         Serial.print("AppSKey:␣");
245         for (size_t i = 0; i < sizeof(artKey); ++i) {
246           if (i != 0)
247             Serial.print("-");
248           printHex2(artKey[i]);
249         }
250         Serial.println("");
251         Serial.print("NwkSKey:␣");
252         for (size_t i = 0; i < sizeof(nwkKey); ++i) {
253           if (i != 0)
254             Serial.print("-");
255           printHex2(nwkKey[i]);
256         }
257         Serial.println();
258       }
259       // Disable link check validation (automatically enabled
260       // during join, but because slow data rates change max TX
```

```
261       // size, we don't use it in this example.
262       LMIC_setLinkCheckMode(0);
263       break;
264     /*
265         || This event is defined but not used in the code. No
266         || point in wasting codespace on it.
267         ||
268         || case EV_RFU1:
269         ||     Serial.println(F("EV_RFU1"));
270         ||     break;
271         */
272     case EV_JOIN_FAILED:
273       Serial.println(F("EV_JOIN_FAILED"));
274       break;
275     case EV_REJOIN_FAILED:
276       Serial.println(F("EV_REJOIN_FAILED"));
277       break;
278       break;
279     case EV_TXCOMPLETE:
280       Serial.println(F("EV_TXCOMPLETE␣(includes␣waiting␣for␣RX␣
             windows)"));
281       // Any data to be received?
282       CheckDownlinkMessage();
283
284       if (Buzzer_flag) {  // If the buzzer flag is active
285         Serial.println(F("Buzzer␣ON"));
286         BuzzerSound(15);      // Make 15 intermitent beeps
287         Buzzer_flag = false;  // Deactivate buzzer flag
288       }
289       // Schedule next transmission
290       os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(
             TX_INTERVAL), do_send);
291       break;
292     case EV_LOST_TSYNC:
293       Serial.println(F("EV_LOST_TSYNC"));
294       break;
295     case EV_RESET:
296       Serial.println(F("EV_RESET"));
297       break;
298     case EV_RXCOMPLETE:
299       // data received in ping slot
300       Serial.println(F("EV_RXCOMPLETE"));
301       // Any data to be received?
302       CheckDownlinkMessage();
```

```
303       break;
304     case EV_LINK_DEAD:
305       Serial.println(F("EV_LINK_DEAD"));
306       break;
307     case EV_LINK_ALIVE:
308       Serial.println(F("EV_LINK_ALIVE"));
309       break;
310     /*
311         || This event is defined but not used in the code. No
312         || point in wasting codespace on it.
313         ||
314         || case EV_SCAN_FOUND:
315         ||    Serial.println(F("EV_SCAN_FOUND"));
316         ||    break;
317     */
318     case EV_TXSTART:
319       Serial.println(F("EV_TXSTART"));
320       break;
321     case EV_TXCANCELED:
322       Serial.println(F("EV_TXCANCELED"));
323       break;
324     case EV_RXSTART:
325       /* do not print anything -- it wrecks timing */
326       break;
327     case EV_JOIN_TXCOMPLETE:
328       Serial.println(F("EV_JOIN_TXCOMPLETE: no JoinAccept"));
329       break;
330
331     default:
332       Serial.print(F("Unknown event: "));
333       Serial.println((unsigned)ev);
334       break;
335   }
336 }
337
338
339 void do_send(osjob_t* j) {
340   /*
341       do_send(osjob_t * j) sends message through LoRaWAN
342
343   j:
344   */
345   // Check if there is not a current TX/RX job running
346   batteryLevel();  // Read battery level
```

```
347   if (LMIC.opmode & OP_TXRXPEND) {
348     Serial.println(F("OP_TXRXPEND, not sending"));
349   } else {
350
351     if (!GPS_flag) {  // If the GPS is not manually activated
352       // Scan surrounding networks
353       n_WIFI = WiFi.scanNetworks();
354       Serial.println("Scan done");
355     }
356
357     if ((n_WIFI <= 1) or GPS_flag) {  // If there is 1 or none
          WiFi or the GPS flag is True
358       Serial.println("Using GPS...");
359
360       digitalWrite(GPS_mosfet, HIGH);  // Turn on the GPS module
361       delay(500);
362       smartDelayGPS(70000);  // Waiting for the GPS to fix as
            many satellites as it can and retrieving the values
363       GPS_flag = false;      // Deactivate the GPS flag
364
365       if ((n_satellites > 0) and ((lat != 0) and (lng != 0))) {
                                                // If the information
           from the GPS is valid
366         sprintf(dataStringUplink, "%01d%2.6f%3.6f%03d",
              n_satellites, lat, lng, BatPercentage);  // Format the
               data according to the Custom Protocol
367         Serial.println(dataStringUplink);
368         // Prepare upstream data transmission at the next
              possible time.
369         LMIC_setTxData2(4, (xref2u1_t)&dataStringUplink, sizeof(
              dataStringUplink), 0);
370         Serial.println(F("Packet queued"));  // Next TX is
              scheduled after TX_COMPLETE event.
371       }
372
373     } else {
374       Serial.println(F("Using WiFi ..."));
375       // Format the info before sending it according to the
            Custom Protocol
376       int channel = 0;
377       for (int i = 0; i <= n_WIFI; ++i) {
378         sprintf(dataStringUplink + (i * 14), "%02X%02X%02X%02X%02
              X%02X%2d%03d", WiFi.BSSID(i)[0], WiFi.BSSID(i)[1],
              WiFi.BSSID(i)[2], WiFi.BSSID(i)[3], WiFi.BSSID(i)[4],
```

```
            WiFi.BSSID(i)[5], -WiFi.RSSI(i)), BatPercentage;
379         channel = i;
380       }
381       // Prepare upstream data transmission at the next possible
            time.
382       Serial.println(channel);
383       Serial.println(dataStringUplink);
384
385       LMIC_setTxData2(channel, (xref2u1_t)&dataStringUplink,
            sizeof(dataStringUplink), 0);
386       Serial.println(F("Packet queued"));    // Next TX is
            scheduled after TX_COMPLETE event.
387     }
388
389     // WiFi.scanDelete();  // Experienced more memory errors
390   }
391 }
392
393
394 void setup() {
395
396   Serial.begin(9600);  // Start the Serial Monitor
397   pinMode(GPS_mosfet, OUTPUT);
398   pinMode(BUZZZER_PIN, OUTPUT);
399   pinMode(vdPin, INPUT);
400   delay(200);
401   Serial.println(F("Starting"));
402   digitalWrite(BUZZZER_PIN, LOW);
403   digitalWrite(GPS_mosfet, LOW);
404
405   WiFi.mode(WIFI_STA);  // station mode: the ESP32 connects to an
        access point
406   WiFi.disconnect();
407   delay(500);
408   delay(500);
409   ss.begin(GPSBaud);
410
411   // LMIC library init
412   os_init();
413   // // Reset the MAC state. Session and pending data transfers
        will be discarded.
414   LMIC_reset();
415
416
```

```
417   // allow much more clock error than the X/1000 default. See:
418   // https://github.com/mcci-catena/arduino-lorawan/issues/74#
        issuecomment-462171974
419   // https://github.com/mcci-catena/arduino-lmic/commit/42da75b56
        #diff-16d75524a9920f5d043fe731a27cf85aL633
420   // the X/1000 means an error rate of 0.1%; the above issue
        discusses using values up to 10%.
421   // so, values from 10 (10% error, the most lax) to 1000 (0.1%
        error, the most strict) can be used.
422   LMIC_setClockError(1 * MAX_CLOCK_ERROR / 40);
423   LMIC_setLinkCheckMode(0);
424
425   // // Start job (sending automatically starts OTAA too)
426   do_send(&sendjob);
427 }
428
429
430 void loop() {
431   // we call the LMIC's runloop processor. This will cause things
         to happen based on events and time. One
432   // of the things that will happen is callbacks for transmission
         complete or received messages. We also
433   // use this loop to queue periodic data transmissions.  You can
         put other things here in the `loop()` routine,
434   // but beware that LoRaWAN timing is pretty tight, so if you do
         more than a few milliseconds of work, you
435   // will want to call `os_runloop_once()` every so often, to
        keep the radio running.
436   os_runloop_once();
437 }
```

List of Figures

1 IoT Wireless Network Communication Protocols [2] . . . 3
2 Sigfox technology based on Ultra-Narrow Band [5] . . . 5
3 Frequency hopping on replicas[5] . . . 6
4 LoRa CHIRPs for SF $\in [7..12]$ and BW $\in [125, 250, 500] kHz$ on the $868.5 MHz$ channel [8] . . . 8
5 Example of a *LoRa* transmission [11] . . . 8
6 LoRaWAN Classes . . . 10
7 Generative design of a piece with different levels of complexity [26] . . . 15
8 Milwaukee Tick™ Tag [39] . . . 19
9 Milwauke ONE-KEY ™ enabled tool [40] . . . 19
10 Milwaukee Tick™ Tag [41] . . . 20
11 DeWalt TOOL CONNECT™ Chip [42] . . . 20
12 Milwaukee Tick™ enabled tool [43] . . . 20
13 DeWalt TOOL CONNECT™ battery [44] . . . 20
14 DeWalt TOOL CONNECT™ Connector [45] . . . 21
15 DeWalt TOOL CONNECT™ Gateway [46] . . . 21
16 Bosch solution . . . 22
17 LILYGO-TTGO-V1.3 PINMAP . . . 26
18 LILYGO-TTGO-V1.3 measurements . . . 26
19 Efficiency vs Output Current TPSM84205EAB [57] . . . 28
20 3D design of the PCB with the modules to check the size . . . 29
21 PCB Schematic . . . 30
22 Footprint top layer . . . 31
23 Footprint bottom layer . . . 31
24 Flowchart with the basic behaviour . . . 34
25 Structure of the payload when sending an uplink message with WiFi data . . . 36
26 Structure of the payload when sending an uplink message with GPS data . . . 36
27 Structure of the payload when sending a downlink message to switch ON or OFF the GPS module and/or the Buzzer . . . 37
28 Battery connector . . . 38
29 Power tool connector . . . 38
30 Scanner setup . . . 39
31 Scanning process using Revo Scan . . . 40
32 Establishing the coordinate system using GOM Inspect . . . 41
33 On the left the mesh obtained from the 3D scan and on the right the solid recreated out of it . . . 42
34 Power tool connector 3D design . . . 42
35 MakerBot Replicator+ . . . 44
36 Form 3 printer . . . 44
37 Washing station . . . 44
38 UV cure station . . . 44
39 First battery connector prototype . . . 45

40 First power tool connector prototype 45
41 Negatives of the board resting inside the UV-curing box 46
42 PCB in the developer 46
43 PCB after been etched 46
44 Final PCB with all the components mounted 47
45 Battery Connector final version with metal connectors 49
46 Power tool Connector final version with metal connectors 51
47 First test using positive plates for AA batteries 52
48 Hand tools used to create the metal connectors 52
49 Male metal connector 53
50 Female metal connector 53
51 Picture of the final assembled prototype 56
52 The message flow in Helium 58
53 Dashboard in *Power BI* when a *GPS* package is received 58
54 Power BI report map showing the gateway coordinates 59
55 Oscilloscope screenshot while measuring the voltage drop in a 1Ω resistor . . 62

List of Tables

1 European ISM duty cycle limitations per band [12]. 11
2 Manufacturer time to fix GPS satellites ∗Time-To-First-Fix (all satellites at -130 dBm) [52] 27
3 Symbol encoding used for the downlink packet 37
4 Current drained by the power tool 60
5 Add-on current drained in different situations 62
6 Skyhook accuracy values in different environments and with different amounts of APs 64

References

[1] "S34 Feasibility study of a monitoring system." DTU inside.

[2] "IoT Wireless Network Communication Protocols." http://emanuelepagliari.it/wp-content/uploads/2020/10/Internet-of-Things-Wireless-Network-Communication-Protocols.png.

[3] "3GPP TR 21.914 Release 14." https://www.3gpp.org/ftp/Specs/archive/21_series/21.914/.

[4] "Sigfox explanation." https://support.sigfox.com/docs.

[5] "Sigfox Technical Overview." https://www.ismac-nc.net/wp/wp-content/uploads/2017/08/sigfoxtechnicaloverviewjuly2017-170802084218.pdf.

[6] "Sigfox payload encryption (5.7)." https://www.aerea.nl/wp-content/uploads/2018/06/Secure-Sigfox-Ready-devices-recommendation-guide-II.pdf.

[7] L. Vangelista, "Frequency shift chirp modulation: The lora modulation," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1818–1821, 2017.

[8] "LoRa Channel Characterization for Flexible and High Reliability Adaptive Data Rate in Multiple Gateways Networks." https://www.mdpi.com/2073-431X/10/4/44.

[9] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[10] "What are LoRa and LoRaWAN?." https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan.

[11] "Figure showing the symbols in a LoRa tranmission." http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html.

[12] "ETSI EN 300 220-2 European duty cycle restrictions." https://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.01_30/en_30022002v030201v.pdf.

[13] "Comparison of prices depending of the protocol." https://www.iotsolutions.com.mt/post/sigfox-vs-lora-vs-nb-iot-who-s-doing-it-best.

[14] "LoRaWAN provider." https://www.helium.com/lorawan.

[15] "The Things Network ecosystem that provides solutions using LoRaWAN." https://www.thethingsnetwork.org/.

[16] "Cibicom main website." https://cibicom.com/.

[17] "GPS consumption." https://www.sparkfun.com/GPS_Guide#:~:text=It's%20a%20lot%20of%20work,around%2030mA%20at%203.3V.

[18] "Active vs passive antenna power drawn." https://www.vectornav.com/resources/inertial-navigation-primer/hardware/rf#:~:text=Passive%20antennas%20consist%20of%20only,current%20draw%20to%20a%20system.

[19] "WiGLE main page." https://wigle.net/index.

[20] "Skyhook main page." https://www.skyhook.com/wifi-location-solutions.

[21] H. M. Yilmaz, M. Yakar, and F. Yildiz, "Digital photogrammetry in obtaining of 3d model data of irregular small objects," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, pp. 125–130, 2008.

[22] M. I. Alba, L. Barazzetti, M. Scaioni, E. Rosina, and M. Previtali, "Mapping infrared data on terrestrial laser scanning 3d models of buildings," *Remote Sensing*, vol. 3, no. 9, pp. 1847–1870, 2011.

[23] "Website from EinScan HX a portable professional laser 3D scanner." https://www.einscan.com/einscan-hx/.

[24] R. H. Kazi, T. Grossman, H. Cheong, A. Hashemi, and G. Fitzmaurice, "Dreamsketch: Early stage 3d design explorations with sketching and generative design," UIST '17, (New York, NY, USA), p. 401â414, Association for Computing Machinery, 2017.

[25] H. Jun and A. Nichol, "Shap-e: Generating conditional 3d implicit functions," 2023.

[26] "Website where the picture of a generative design was extracted." https://www.plm.automation.siemens.com/global/en/our-story/glossary/generative-design/27063.

[27] R. Waghmare, D. Suryawanshi, and S. Karadbhajne, "Designing 3d printable food based on fruit and vegetable productsâopportunities and challenges," *Journal of Food Science and Technology*, vol. 60, no. 5, pp. 1447–1460, 2023.

[28] Z. Jin, Y. Li, K. Yu, L. Liu, J. Fu, X. Yao, A. Zhang, and Y. He, "3d printing of physical organ models: recent developments and challenges," *Advanced Science*, vol. 8, no. 17, p. 2101394, 2021.

[29] F. Bandinelli, L. Peroni, and A. Morena, "Elasto-plastic mechanical modeling of fused deposition 3d printing materials," *Polymers*, vol. 15, p. 234, Jan 2023.

[30] G. M. Fortunato, M. Nicoletta, E. Batoni, G. Vozzi, and C. De Maria, "A fully automatic non-planar slicing algorithm for the additive manufacturing of complex geometries," *Additive Manufacturing*, vol. 69, p. 103541, 2023.

[31] "MakerBot Tough filament." https://www.makerbot.com/3d-printers/materials/tough/.

[32] "Tough 2000 resin." https://formlabs.com/store/materials/tough-2000-resin/.

[33] "Next PCB manufacturer of printed circuit boards." https://www.nextpcb.com/pcb-assembly-quote#/pcb-assembly-quote.

[34] S. Jeong, D.-H. Kim, J. Song, H. Kim, S. Lee, C. Song, J. Lee, J. Song, and J. Kim, "Smartwatch strap wireless power transfer system with flexible pcb coil and shielding material," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 5, pp. 4054–4064, 2019.

[35] P. Aqueveque, R. Osorio, F. Pastene, F. Saavedra, and E. Pino, "Capacitive sensors array for plantar pressure measurement insole fabricated with flexible pcb," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4393–4396, 2018.

[36] A. Petropoulos, D. N. Pagonis, and G. Kaltsas, "Flexible pcb-mems flow sensor," *Procedia Engineering*, vol. 47, pp. 236–239, 2012. 26th European Conference on Solid-State Transducers, EUROSENSOR 2012.

[37] "JLCPCB world leading company in PCB production." https://jlcpcb.com/.

[38] "Apple AirTag main website." https://www.apple.com/airtag/.

[39] "Website of the Milwaukee tag." https://www.milwaukeetool.com/Products/48-21-2301.

[40] "Website of the Milwaukee tool." https://www.milwaukeetool.com/2705-20.

[41] "Website of the DeWalt TOOL CONNECT™ tag." https://www.dewalt.com/product/dce041/tool-connect-tag-single.

[42] "Website of the DeWalt TOOL CONNECT™ chip." https://www.dewalt.com/product/dce042/tool-connect-chip.

[43] "Website of the DeWalt TOOL CONNECT™ tool." https://www.dewalt.com/product/dcf888p2bt/20v-max-xr-brushless-cordless-impact-driver-tool-connect-kit?tid=.

[44] "Website of the DeWalt TOOL CONNECT™ battery." https://www.dewalt.com/product/dcb205bt/20v-max-tool-connect-5ah-battery?tid=.

[45] "Website of the DeWalt TOOL CONNECT™ connector." https://www.dewalt.com/product/dce040/tool-connect-20v-max-connector.

[46] "Website of the DeWalt TOOL CONNECT™ gateway." https://www.dewalt.com/product/dce081/construction-asset-gateway.

[47] "Heltec's WiFi LoRa 32 (V3) device." https://heltec.org/project/wifi-lora-32-v3/.

[48] "Lilygo TTGO LoRa32 V1.3 device." http://www.lilygo.cn/prod_view.aspx?TypeId=50003&Id=1253&FId=.

[49] "ESP32 chip used in LilyGo board." https://www.semtech.com/products/wireless-rf/lora-connect/sx1276.

[50] "Semtech sx1276 LoRaWAN module." https://www.semtech.com/products/wireless-rf/lora-connect/sx1276.

[51] "Molex ISM 105262 flexible LoRa antenna." https://dk.rs-online.com/web/p/wifi-antenner/8197398.

[52] "GNSS module Ublox NEO-7N." https://content.u-blox.com/sites/default/files/products/documents/NEO-7_DataSheet_%28UBX-13003830%29.pdf.

[53] "The same module was not found anywhere so information was obtained from a blog." https://www.hackster.io/infoelectorials/review-009-ublox-neo-7n-gps-module-review-518dd5.

[54] "Datasheet of the antenna of the GNNS module." https://abracon.com/patchantenna/APAE1575R2520ABDD7-T.pdf.

[55] K. Yao, M. Ye, M. Xu, and F. Lee, "Tapped-inductor buck converter for high-step-down dc-dc conversion," *IEEE Transactions on Power Electronics*, vol. 20, no. 4, pp. 775–780, 2005.

[56] M. Zhao, M. Li, S. Song, Y. Hu, Y. Yao, X. Bai, R. Hu, X. Wu, and Z. Tan, "An ultra-low quiescent current tri-mode dc-dc buck converter with 92.1x0025; peak efficiency for iot applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 1, pp. 428–439, 2022.

[57] "DC-DC step down regulator datasheet." https://www.ti.com/lit/ds/symlink/tpsm84205.pdf?ts=1686513788391.

[58] "SWIFT proprietary buck design from Texas Instruments." https://www.ti.com/lit/sg/slvt165g/slvt165g.pdf?ts=1687445026447&ref_url=https%253A%252F%252Fwww.google.com%252F.

[59] "1800 mAh LiPo battery." https://dk.rs-online.com/web/p/genopladelige-batterier-i-specialstoerrelser/1449405?cm_mmc=DK-PLA-DS3A-_-google-_-CSS_DK_DK_Batterier_og_opladere_Whoop-_-(DK:Whoop!)+Genopladelige+batterier+i+specialst%C3%B8rrelser-_-1449405&matchtype=&aud-827186183886:pla-333331382340&gclid=Cj0KCQjw7uSkBhDGARIsAMCZNJsU0AJvShZ90mRN9Suq3Jl4wX1BhERD9n7NpNJwPLElJP_qTaB4P8EaAv_BEALw_wcB&gclsrc=aw.ds.

[60] "Diode 1N4001 datasheet." https://www.digikey.dk/en/products/detail/diodes-incorporated/1N4004-T/604.

[61] "Revopoint product comparison." https://www.revopoint3d.com/products-compare/.

[62] "Revopoint mini manual." https://wwww.revopoint3d.com/wp-content/uploads/download/Revopoint%20MINI%20User%20Manual.pdf?_ga=2.242374722.1333139885.1677080146-1820728515.1675244845.

[63] "Reverse Engineering from a 3D Scan." https://www.youtube.com/watch?v=imGrla3b3Mo.

[64] "MakerBot Replicator +." https://www.makerbot.com/es/3d-printers/replicator/.

[65] "Website of the Original Prusa i3 MK3S+." https://www.prusa3d.com/product/original-prusa-i3-mk3s-3d-printer-3/.

[66] "Formlabs Form 3+." https://formlabs.com/3d-printers/form-3/tech-specs/.

[67] "Battery connector business." https://www.amphenol-cs.com/custom-battery-connectors.

[68] "Buy LilyGo TTGO V1.3." https://www.aliexpress.com/item/4000628100802.html.

[69] "Price of the DC-DC regualtor." https://www.digikey.dk/da/products/detail/texas-instruments/TPSM84205EAB/7561620.

[70] M. Catena, "arduino-lmic." https://github.com/charlespwd/project-title, 2013.

[71] H. Luan, L. Tianze, Z. Xia, and J. Chuan, "Test of battery internal resistance," vol. 276, p. 012166, feb 2011.

[72] "Website of the U-center, GNSS evaluation software for Ublox devices." https://www.u-blox.com/en/product/u-center.

[73] “Website of Postman an API platform.” https://www.postman.com/.