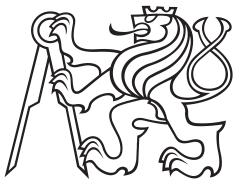
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Measurement

Acoustic Event Detection System

Bc. Michal Opočenský

Supervisor: prof. Ing. Jan Holub, Ph.D.
May 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Opočenský** Jméno: **Michal** Osobní číslo: **465946**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Systém pro detekci akustických událostí

Název diplomové práce anglicky:

Acoustic Event Detection System

Pokyny pro vypracování:

Navrhněte a realizujte systém pro detekci a lokalizaci akustických pulsních událostí. Systém musí obsahovat nejméně 3 čidla, komunikující prostřednictvím vhodné zvolené bezdrátové technologie (Wifi, LTE, LoRa apod.). Při řešení využijte teoretické poznatky a SW řešení serveru, nabyté v rámci týmového projektu. Funkčnost systému demonstrujte experimentem, odhadněte spolehlivost detekce a spolehlivost lokalizace.

Seznam doporučené literatury:

[1] Hakl J.: Jednotka pro akustickou detekci, diplomová práce 2019 FEL ČVUT, dostupné z https://dspace.cvut.cz/handle/10467/80115
[2] https://github.com/jurasofish/multilateration/
[3] Svatoš, J., Holub, J.: Smart Acoustic Sensor, IEEE RTSI 2019, DOI: 10.1109/RTSI.2019.8895591

Jméno a pracoviště vedoucí(ho) diplomové práce:

prof. Ing. Jan Holub, Ph.D., katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **15.01.2021** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce:
do konce letního semestru 2021/2022

prof. Ing. Jan Holub, Ph.D.	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Acknowledgements

I would like to thank Ing. Jakub Svatoš, Ph.D for his valuable insight into given topic and never-ending optimism. Many thanks to the guys that kick started this work on the team project, namely Bc. Jiří Minarik, Bc. Michal Špaček, Bc. Josef Čech and Bc. Pert Vanc. Big thanks to my family for grammatical correction of the thesis. And finally, thanks to Bc. Kristýna Kučerová for providing tikz-hotline service.

Declaration

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

V Praze, květen 2021

“I declare that this work is all my own work and I have cited all sources I have used in the bibliography in accordance with the Methodological Instruction on observance of ethical principles in the preparation of university theses.”

Prague, May, 2021

. . . . . .

Abstract

The main goal of the thesis is to create a system capable of gunshot detection, classification, and localization. The detection system consists of a specialized board and a server application.

At first, the gunshot detection algorithm is executed for filtration of non-gunshot events. Afterwards, the features are extracted by Mel-Frequency Cepstral Coefficients. The feature vector is then passed to the gunshot classification, performed through Support Vector Machine.

The localization task is executed on precisely timestamped acoustic events that are coupled with position of the measuring devices (nodes) on the server. The aggregated data are utilized for solving the Time of Arrival Localization Problem. Two different methods are described based on the number of nodes that detected the event.

The created server application solves the localization task as mentioned above but also offers visualization and administration of users, nodes, and node's messages.

The proposed board is able to acquire position with precise timestamping and send the required information through LoRaWAN network to the server. The board implements detection and classification algorithms and also offers a command line interface for setting the firmware's parameters such as detection algorithms' coefficients.

Keywords: gunshot detection, gunshot classification, embedded, server application, multilateration, Time of Arrival localization

Supervisor: prof. Ing. Jan Holub, Ph.D.
A3-320, Technická 2, Praha

Abstrakt

Hlavním cílem této práce je vytvořit systém schopný detekce, klasifikace a lokalizace střelby. Systém se skládá z dedikované desky a serverové aplikace.

Systém detekuje zvukové události a jako první krok vyfiltruje události, které nejsou střelba. Následně jsou klíčové vlastnosti nahrávky extrahovány pomocí Mel-Frequency Cepstral Coefficients. Na vektoru klíčových vlastností je dále provedena klasifikace použitého kalibru zbraně, kterou provádí metoda podpůrných vektorů (Support-Vector Machine).

Lokalizace střelby je prováděna na zvukových událostech, ke kterým je připojena velmi přesná časová značka (timestamp) a pozice měřícího přístroje (uzlu). Data shromážděná z jednotlivých zařízení jsou použita pro řešení lokalizačního problému na základě změřeného času zaznamenání (Time of Arrival Localization Problem). Pro jeho řešení jsou popsány dvě různé metody, lišící se dle počtu měřících zařízení, které danou událost detekovaly.

Vytvořená serverová aplikace je nejen schopna řešit lokalizační úlohu popsanou výše, ale také poskytuje vizualizaci s administrací uživatelů, uzlů a zpráv uzlů. Navržená deska je schopna získat svou pozici spolu s přesnou časovou značkou události a odeslat všechny potřebné informace pomocí LoRaWan sítě na server. Na desce je naimplementován jak detekční, tak klasifikační algoritmus. Navíc deska nabízí rozhraní ve formě příkazové řádky pro nastavení parametrů aplikace, jako jsou například koeficienty detekčního algoritmu.

Klíčová slova: detekce střelby, clasifikace střelby, embedded, servrová aplikace, multilaterace, Time of Arrival lokalizace

Překlad názvu: Systém pro detekci akustických událostí

Contents

Figures

Tables

Chapter 1
Introduction

Nowadays, the sensors are lightweight, power efficient and can be easily placed in public areas. There are already surveillance cameras around the city that help solve crimes. In the same manner, seismic sensors can be placed to predict incoming earthquakes and therefore prevent loss of life. Strategically placed sensors can also monitor other events, such as sound spikes. The audio spikes can signal a car crash, gunshot, or other dangerous event.

With the rapidly evolving Internet of Things (IoT), combining the detection of sound waveforms with a classification algorithm for, for example, gunshots, is just another logical step. In the thesis, the usage of sound recording devices for the detection and localization of audio events is examined. The events can be afterwards automatically classified, which might be used for the detection of dangerous states.

One part of the thesis is devoted to a real implementation from scratch of the algorithm for gunshot detection described in the paper [1] created on the CTU FEL department of Measurement.

The thesis is complemented by the proposition and implementation of algorithms for classification of gunshot caliber and localization based on time of arrival (TOA).

The algorithms for detection and classification are deployed on the hardware board and tested. The last algorithm for localization was implemented and tested on the server.

The work is divided into three main parts: Board, Server and Algorithms.

In the Board Chapter 2, the elaborated creation of the system node can be found. The nodes detect gunshots, classify the audio recording, and the results are then sent to server, see the Figure 1.1.

In the Server Chapter 3, the features and capabilities of the resulting server application are specified. The server processes the received data and then solves the localization task.

In the Algorithms Chapter 4, the detection and classification algorithm in the board firmware and the localization task in the server application are described. The classification algorithm is used for the classification of recorded audio. It uses Support Vector Machine classifier with Mel-Frequency Cepstral Coefficients feature extraction. The detection algorithm is performed in the time domain and uses median filters. The localization algorithm uses Levemberg-Marquardt iterative solution of a set of nonlinear equations with possible coordinate system transformation from 3D to 2D.

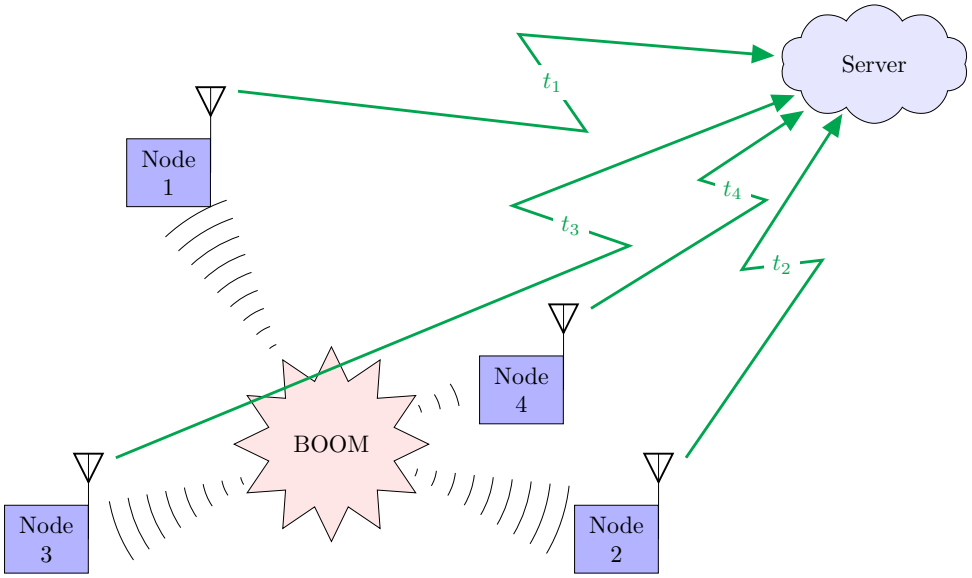


Figure 1.1: Project structure - nodes send messages (timestamps) based on events to the server.

The results for each described method are placed within the corresponding part, and the thesis summary is provided in Chapter 5.

1.1 State of the art

At this time, providers of gunshot localization and detection services already exist, e.g., the U.S. ShotSpotter [2] or V5 Acoustic Gunshot Sensor from the V5 Systems [3]. Currently, these services use human evaluation step for the final gunshot classification.

Many papers focus on localization in situations where a low number of sensors are available, therefore the gunshot is recorded from a greater distance (in kilometers) [4]. This localization is useful, for example, for sniper localization, however the open-field environment is considered. There is research for the urban area, where the number of sensors can be large, scattered through the city structure.

1.2 Methods

The detection of the gunshot can be done in three different ways: optical flash detection, muzzle blast sound wave impulse, and shock wave created by a traveling bullet.

The localization of a Muzzle blast impulse can be obtained via two methods: Time of Arrival, or with an array of microphones. If the Time of Arrival is used, just one microphone is sufficient and at least 3 sensors are necessary. On the other hand, the array of microphones requires at least three microphones. From the time delay between microphones, the direction of the incoming gunshot is computed and from at least two directions, the position can be acquired (azimuth and elevation). The array can be also used for detection using shock wave

In this work, the muzzle blast sound wave impulse (Sections 4.1, 4.2) is used for the detection and the Time of Arrival method (Section 4.3) is utilized for the localization.

1.3 Requirements

The resulting system shall be capable of localization and classification of gunshots with the Time of Arrival method of localization.

The hardware shall be powerful enough for further development of possibly more demanding computations.

Every node shall listen for the acoustic event. When an acoustic event occurs, it is precisely timestamped and classified, whether it is a gunshot or not. If the acoustic event is a gunshot, the timestamp with the position of the node is sent to the server.

The server aggregates the incoming data from the nodes and performs a localization algorithm. The server shall provide a user and administration interface.

The requirements specified imply the top level structure of the project in the form shown in the Figure 1.1.

Chapter 2

Board

The board is in the naming convention of the server called a node, which is one of the basic parts of the system chain.

The design of the node was created based on the defined requirements:

1. Position acquisition
2. Precise timestamping of an event
3. Radio communication
4. Temperature measurement
5. Battery operation
6. Audio recording
7. Storing parameters
8. USB communication
9. Floating Point Unit (FPU)
10. Mass-space storage (SD card)

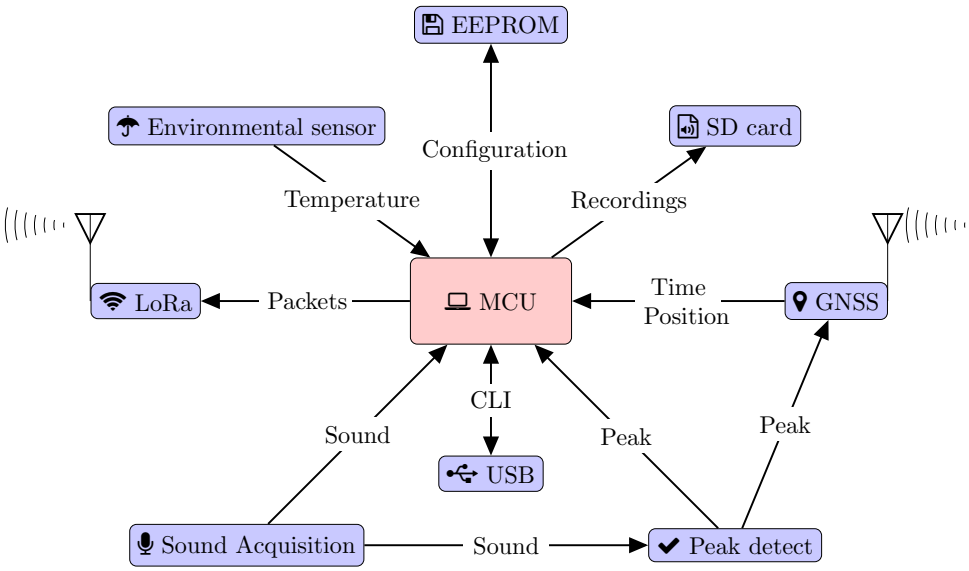


Figure 2.1: Board components interconnection schema.

2.1 Components

Components determine not only the capabilities of the board but also the ergonomics of the firmware programming, reliability, etc. For board's components logical interconnection schema see the Figure 2.1.

2.1.1 Micro-Controller

The used microcontroller is from the family stm32, manufactured by ST. The choice was made based on my preceding work with API of the provided High Abstraction Layer (HAL) libraries.

The next step was the selection of the right type. The ARM Cortex M0+ is not suitable because it does not contain FPU, moreover it is not powerful enough for more computationally demanding tasks. We selected a chip with higher performance than estimated for the proposed application to allow further functionality expansion. Furthermore, the chip should have enough flash and RAM for possible development of more resource demanding algorithms such as classification using deep neural network. Due to the limited availability of other chips the chip stm32F413RH [5] was selected.

The stm32F413RH microprocessor satisfies the requirements (6–10). Main parameters are:

- 320 kB of SRAM
- 1.5 MB of Flash memory
- Clock frequency up to 100 MHz
- FPU
- Real Time Clock (RTC)
- Internal 12-bit ADC
- Power consumption of 112 µA MHz^{-1} (peripherals off)
- SDIO interface (for SD card)
- USB 2.0
- Supply voltage 1.7 V up to 3.6 V

The SDIO interface allows connection with an SD card, which can be used as mass storage.

2.1.2 LoRaWAN

The LoRa, developed by Semtech, is a low-power communication network that specifies only the physical layer, LoRaWAN defines the higher layers of the communication.

The LoRa in Europe is using a free license radio band on 868 MHz. As there is no fee for using the band, we can use any number of devices that we need. LoRa clients should comply with solidarity rules to enable the coexistence of more devices in the range.

LoRa uses Chirp spread spectrum modulation. The distance between a LoRa gateway and the device can be up to 10 km based on the radio power, antennas, and environmental conditions.

At first, the chip Semtech SX1278 was considered but was not finally used, because the LoRaWAN stack needs to be run on the MCU and libraries for the LoRaWAN stack from Semtech, which are not well documented. Therefore, the Microchip RN2483 was selected, because it provides a simple text-based communication protocol over UART, where the LoRaWAN stack is running on the chip.

The LoRaWANs maximal packet size depends on the transmission speed; at the slowest data rate, it is 52 B.

In the specification of LoRaWAN, there is also an option for the capability of localization. The localization is done in such a way that if a message is received by a gateway, it is precisely timestamped with GPS and sent to the processing server that gathers the positions of the gateways with timestamps.

If the message is received by multiple gateways, the timestamps and gateways positions are used in localization algorithms as described in Section 4.3 and [6].

The LoRaWAN has two authentication methods, Activation By Personalization (ABP) and Over-The-Air Activation (OTAA). With the ABP authentication method, the device has all keys and device address hardcoded, this method is simpler because the devices do not need joining to the network.

On the other hand, the OTAA authentication method has a joining phase with a dynamic assignment of security keys to the device address.

For the OTAA authentication method, the device needs to store Device EUI (8 B), application EUI and application key (16 B). The ABP requires device EUI, application EUI, device address, network session key, and application session key.

The LoRaWAN devices usually have additional protection based on frame counting. The frame counter is automatically incremented when a message is received by the server, and compared with the frame counter contained in the message. If the counters mismatch, the incoming message is not accepted. This feature was disabled for development purposes because the board resets the frame counter during every restart.

The LoraWAN has three device classes based on communication profiles:

Class A: after transmitting a message, there are two receiving windows delayed specific times (1 s, 2 s). The rest of the time it is sleeping. This class is the most power-saving mode. Example of communication is visible in Figure 2.2.

Class B: is the same as for class A but in addition has receiving windows time-synchronized with a gateway.

Class C: extends the class A with a receiving window opened permanently unless transmitting.

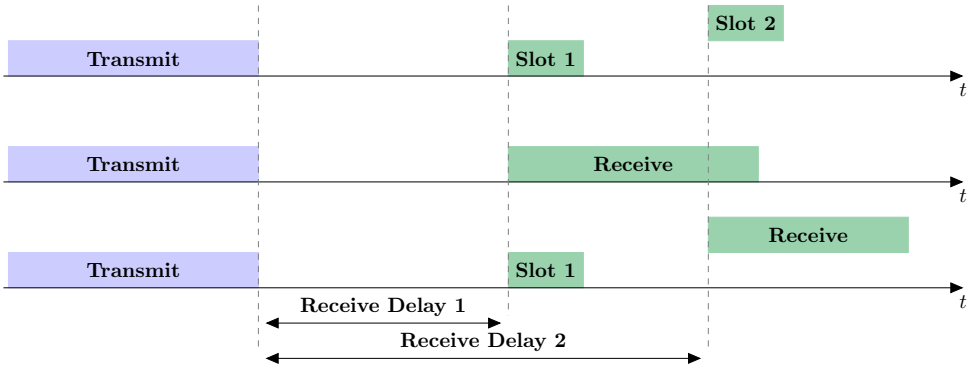


Figure 2.2: LoRaWAN device class A timing diagram, all possible cases of receiving message, first from top: no message, second: message received in first window, third: message received in second window.

LoRaWAN was selected based on extreme low power consumption and very long distance communication. It is suitable for a small amount of data like event definition with timestamp and position. If a complete recording of the acoustic event is to be sent, a different network type for transferring a bigger amount of data has to be used.

Other Networks

Apart from LoRa, there are other types of networks with various advantages and disadvantages:

WiFi is a high power with a high data throughput network.

Bluetooth offers low power mode, but the range of the network is very limited.

NarrowBand can provide low power with small messages. This network is commercial and each message is charged by a local service provider based on an agreement.

GSM (LTE) has the largest coverage from all mentioned networks, but requires an agreement with a service provider similar to the Narrowband network.

Providers

In the Czech Republic, there are two providers CRA (České Radiokomunikace) [7] or The Things Network (TTN) [8]. CRA offers an enterprise-grade network with dense coverage in comparison to TTN that is a community-driven network, so the gateways are not as reliable as from CRA.

It is also possible to create your own network, for this purpose exists open-source project ChirpStack [9].

Gateway

The area, where the board is developed, is not covered by the chosen network TTN, so a new own gateway had to be created for the development. The gateway is based on raspberry PI 3 with IC880a LoRaWAN concentrator board, shown in Figure 2.3. The gateway was connected and registered to TTN using manual [10].

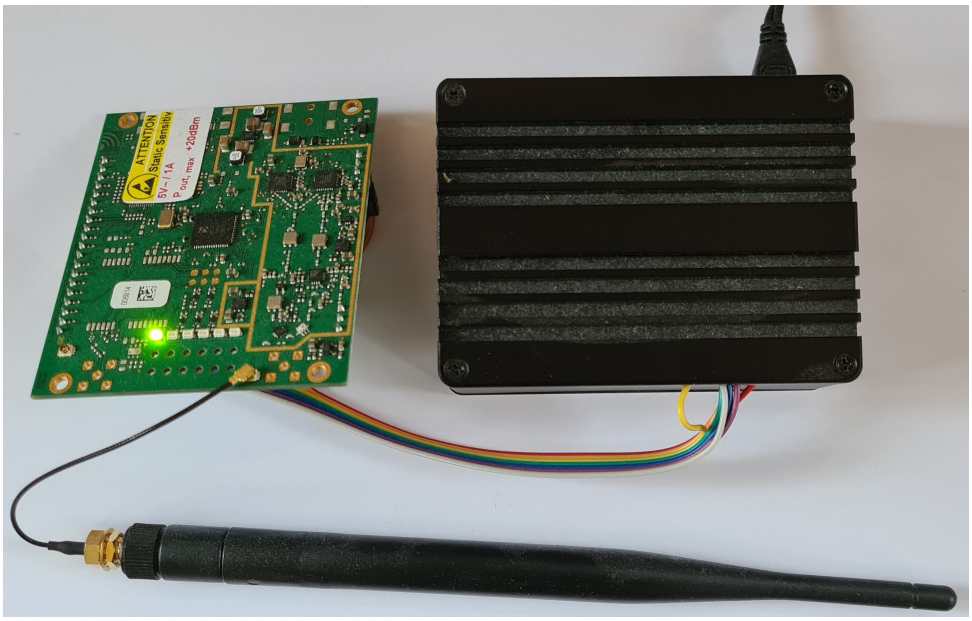


Figure 2.3: Photo of homemade LoRaWAN gateway.

2.1.3 Position and Timestamp

Acquisition of position and precise timestamping is provided with GPS from Ublox professional grade, model LEA-6T. The GPS is capable of timestamping an event with the accuracy of RMS 30 ns.

The communication interface with MCU is UART. USB communication interface is also provided with custom configuration and testing software on Windows with the name u-center [11].

Declared position accuracy is 2.5 m CEP. The supply voltage is 2.7 V to 3.6 V.

Communication protocols

The GPS is capable of communication in two protocols at the same time. The user can select in the configuration the NMEA messages, USBX protocol, or both.

NMEA protocol [12] is a standardized ASCII based message protocol, where each message starts with $ and ends with <CR><LF> (Carriage return, Line feed). Every message contains two hexadecimal ASCII characters as a security checksum which is created with XOR function. The message structure is shown in Figure 2.4.

```
$GPGLL,4717.11634,N,00833.91297,E,124923.00,A,A*6E
```

Listing 2.1: An example of NMEA position message

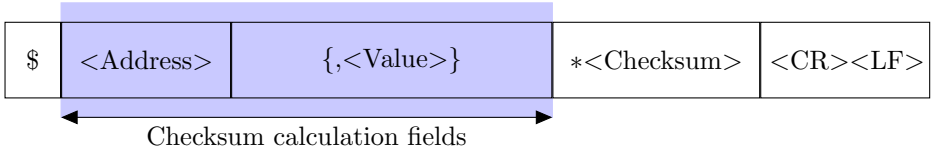


Figure 2.4: NMEA message structure.

UBX protocol [13] is uBlox proprietary byte-oriented protocol, where messages start with two bytes synchronizing characters followed by byte for packet class specification, message ID, two bytes for data length, data and at the end are two bytes for security checksum. The message structure is in Figure 2.5.

```
{ 0xB5, 0x62, 0x0D, 0x03, 0x00, 0x00, 0x10, 0x3D }
```

Listing 2.2: An example of UBX timestamp polling packet

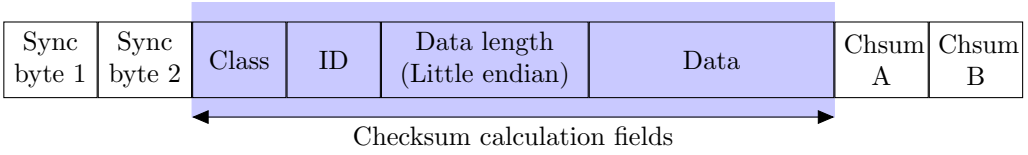


Figure 2.5: UBX packet structure.

2.1.4 Battery

The board shall be capable of independent operation on the battery for at least 4 hours during an external power interruption. For the power backup operation was selected a single cell high drain Li-Ion rechargeable battery IMR 18650, with nominal voltage 3.7 V and capacity 9.25 W h.

Battery Charger

For charging a battery during a standard external power, a battery-friendly charger for Li-ion single-cell battery from Texas Instruments BQ24075TRGTR is used based on previous positive experience.

The charger has internal temperature protection and a maximal charging current limiter. The maximal and minimal charging cutoff temperature can be set via an external resistor. With resistors can be also set the maximal charging current. The profile of current during charging of the battery is plotted in Figure 2.6.

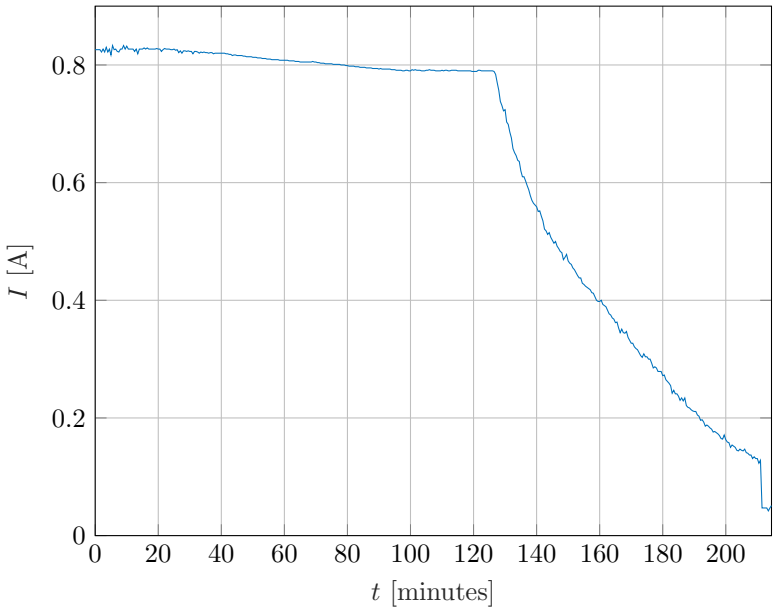


Figure 2.6: Battery charging current profile.

Battery Monitor

If the device is capable of battery-powered operation, it is quite convenient to be able to get the state of the battery, like percentage. For this purpose, there exists ModelGauge Battery Fuel Gauge Technology from Maxim Integrated [14]. With this particular technology in mind, the chip MAX17260 was selected. The chip can learn by itself all necessary parameters such as capacity and state of charge from the battery usage. It can also estimate the life expectancy of the battery by monitoring the capacity.

2.1.5 Nonvolatile Memory

Nonvolatile memory keeps stored information even in case of losing power. The MCU has two types of memory: volatile RAM, where the program runs, and nonvolatile flash, where the program is stored. Flash memory has to be altered in blocks, which means that the change of just one byte requires reading the whole block, storing it inside RAM, modifying it as desired,

erasing the original block in the flash, and storing the modified block back to the flash. It is a complicated process and in case of error, the program can be modified.

Due to the issues described, the external nonvolatile EEPROM memory was added. The selected nonvolatile memory from ST, M24C16 which is 16 kB EEPROM memory with I^2C interface. The chip was chosen based on previous experience. The memory is divided into independent eight pages where every page has its own I^2C address. Its operations are byte-oriented.

2.1.6 Environment Sensor

For the capability of temperature measurement, the sensor from Bosch was selected. The sensor BME280 is capable of measuring not only temperature but also atmospheric pressure and humidity. The selected chip is an automotive grade product with a well-documented official C library on GitHub [15].

2.1.7 Sound Acquisition

The audio recording is one of the main features of the board and can be accomplished with an external or internal Analog-Digital Converter (ADC). The internal ADC is 12 bit and can be sampled without interrupting MCU via timers and Direct Memory Access (DMA). In addition, external 16 bit ADC with SPI interface from Texas Instruments, ADS8866, was selected.

The signal needs to be amplified before conversion from the microphone, for amplification the operational amplifier (op-amp) from Microchip MCP6002 is used.

The external ADC and op-amp were selected based on [16].

The resolution of internal ADC (12 bit) is sufficient because we are looking for step changes instead of detail during the signal. To process a signal with more than a 16 bit ADC, more CPU power and especially more space for data storage is needed.

The selected sampling frequency of sound acquisition is 44 776 Hz. It is the closest higher frequency to 44.1 kHz, the standard CD sample rate, achievable with timer configuration registers. The created sampling frequency deviation is 676 Hz which relatively is 1.5 %.

Peak Detection

The function of the peak detector is to create an event based on a recognized signal [16]. The event then triggers the GPS module for timestamping and starts the digital signal processing (DSP).

The peak detector contains a standard peak detection circuit with forgetting and a high-pass filter, shown in Figure 2.7. The peak detection circuit consists of the diode and a capacitor, where the diode permits discharging of the capacitor when a lower voltage is on the input. The relation between input and output is depicted in Figure 2.8. The output from the peak detection is connected to the op-amp with gain $\frac{120}{51} + 1 \approx 3.35$ to ensure a true logical value on the output.

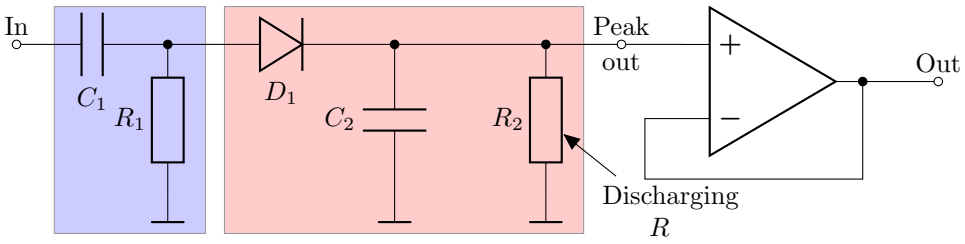


Figure 2.7: Peak detector logical schema, blue is high-pass filter, red is the peak detection with discharging resistor and on output is op-amp with gain.

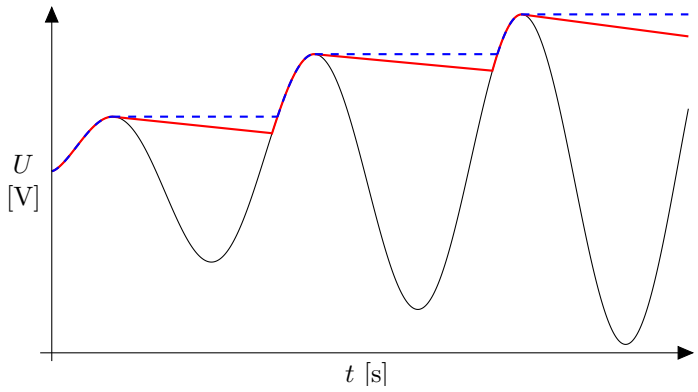


Figure 2.8: Peak detector signal processing, black is the input signal, red is the output signal with discharging R, blue is the output signal without discharging (ideal peak detector).

2.2 Board Preparation

The Board schema and Printed Circuit Board (PCB) layout were created in the software Altium Designer [17]. The Altium Designer is a professional tool for designing boards.

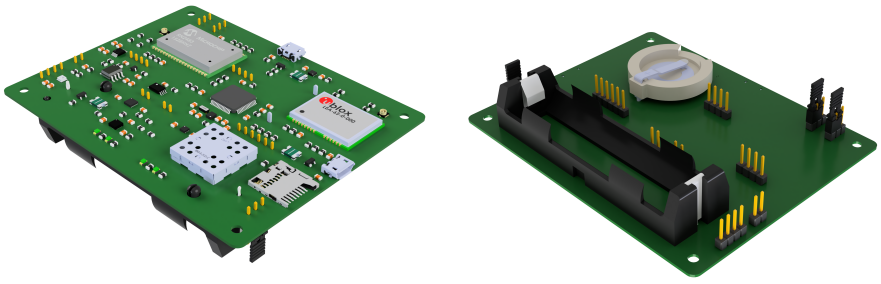
2.2.1 Board Schema

To create a wiring diagram of the selected components, it was necessary to specify power supplies. The Texas Instrument LMZ20502 switching power supply was selected for the digital part. The switching power supply is enclosed inside a shielding cage for the improvement of Electro-Magnetic Compatibility (EMC). A Texas Instruments LP5912-3.3 linear voltage regulator was selected for the analog part of the power supply, especially for the audio amplifier and ADC.

For security reasons, reversible 1.5 A fuses are added for every input power source.

A micro USB port was selected for configuration and battery charging. An additional UART interface is provided internally in case the integrated LoRaWAN does not meet the communication requirements and it is necessary to add other network device (e.g. GSM / LTE).

Another safety measure is an antistatic ring with the exposed ground around the edge of the board. It enables manipulation of the board without exposing the board to static electricity. The final design of PCB contains four layers: two layers with logical connections, one grounding layer (ground plane), and one layer for power supplies only (power plane). Figure 2.9 shows top and bottom side visualization. For maximum EMC is in free areas of the logical layers poured out the ground with stitching.



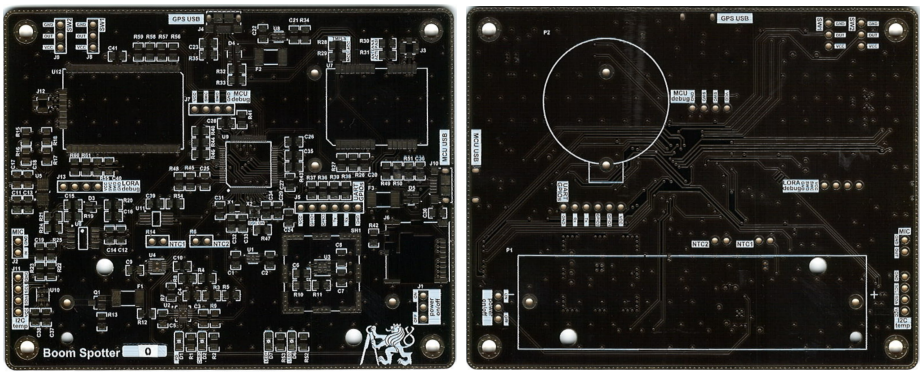
(a) : Top side. **(b) :** Bottom side.

Figure 2.9: Board visualization.

After the successful deployment of the selected components and the interconnection of all connections, step files for the production of the PCB and schema in Chapter A were generated with the Altium Designer software.

2.2.2 PCB Mounting

The PCB was manufactured by Printed s.r.o. (shown in Figure 2.10) and the components were mounted in home condition (photo of finished board is in Figure 2.11). Several components such as charger, switching power supply, or battery monitor are in QFN housing with a thermal pad which requires an oven or rework station for soldering.



(a) : Top side. **(b) :** Bottom side.

Figure 2.10: Board PCB after manufacturing.

Probably due to the inability to comply with the temperature soldering profile, it was not possible to install the MAX17260 battery monitoring chip in home conditions even with a rework station without damage.

In some cases (especially with the LoRa chip) there was a problem with via (an electrical connection between layers) masking. LoRa chip has no masking on the bottom side and after mounting on the board there was a short circuit with connections that lead under the chip. In these cases, it was necessary to stick the Kapton tape to the board for vias insulation.

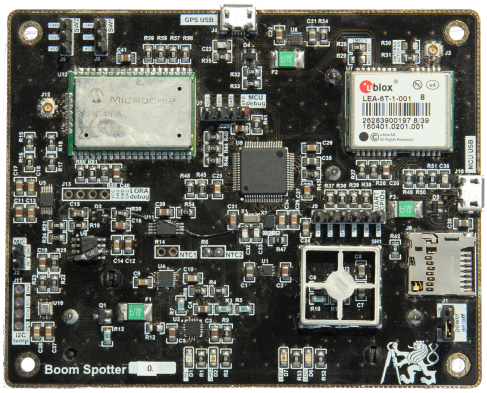
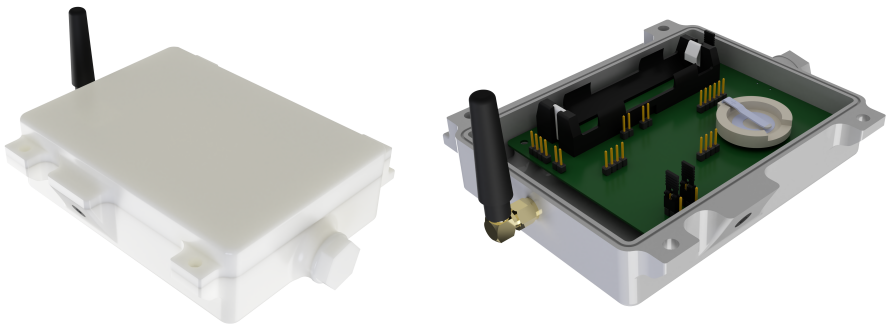


Figure 2.11: Fully mounted board, top side.

2.2.3 Case for Board

A tailor-made box was created for testing in outdoor conditions, shown in Figure 2.12. The design of the box was prepared in 3D software OnShape [18] and then printed at home on a 3D printer AnyCubic Photon resin printer. The threaded plug visible in Figure 2.12a is used for device configuration and battery charging.

Intensive testing with the created box has shown that it is more suitable to use a generic waterproof box for electrical installations. Commonly used boxes do not attract so much attention and have higher long-term weather resistance.



(a) : Closed box. **(b) :** Opened box.

Figure 2.12: Custom case for board.

2.3 Firmware

The firmware, which is the program of the main STM32 processor, defines the behavior of the whole system. It connects individual components on the board to each other based on logical links. The functionality of the board is shown in Figure 2.13. The firmware operates components from the lowest level via registers (timer interrupt), via digital communication on the bus (environment sensor), to communication with GPS and network communication via LoRaWAN.

It was developed in Atolollic TrueSTUDIO [19] due to the previous experience with the toolchain.

In the code, the naming convention is that the name of each variable starts with a data type specification using, `in_` (for input) or `out_` (for output) variables in funciton calls, followed by a sequence of letters standing for the type of variable. For example `out_pai16SoundData` stands for the variable which is output from the function and its type is a pointer to the array of `int16_t` (`int16_t *SoundData[]`).

The convention prevents unwanted implicit data type conversion. It also helps with specification, if the pointer in the function call is an input or the output argument. The modification `const` could solve this problem, but it is not used for example in STs HALs. The code is documented using comments in code and Doxygen [20].

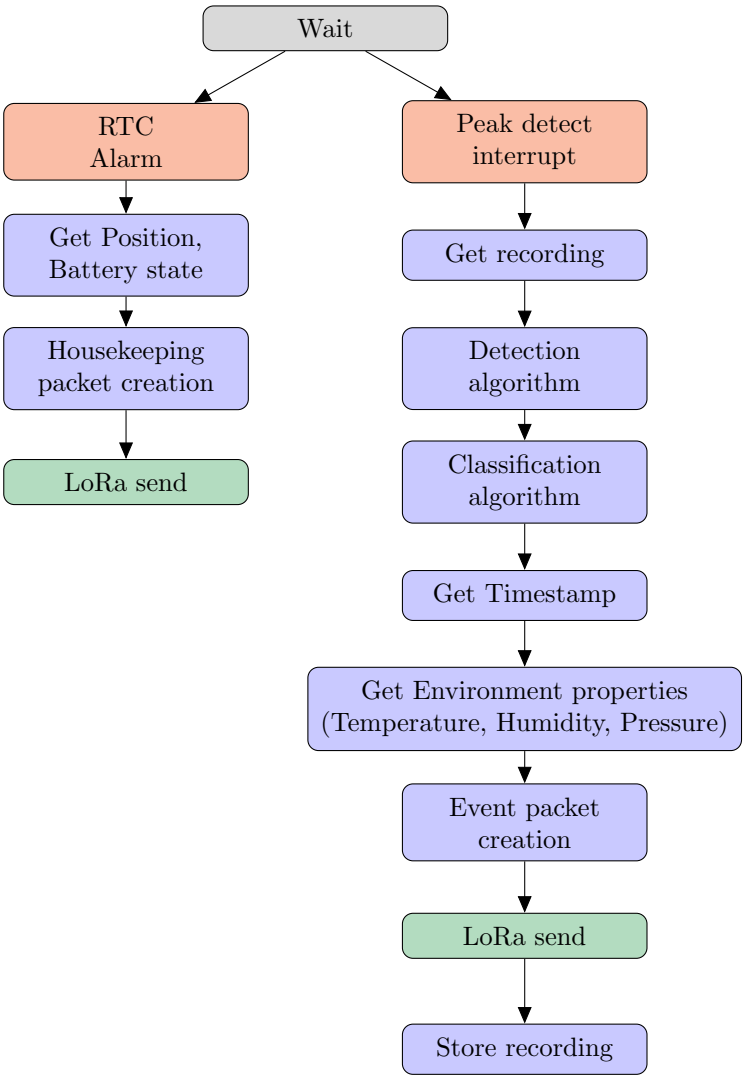


Figure 2.13: Board functionality flowchart.

2.3.1 Firmware Structure

The firmware is internally divided into two parts, where each part consists of folders. The division is outlined in Figure 2.14. The first part is the written code and the second part is auto-generated code via CubeMX software [21] from ST.

The written code contains the following parts:

BSP is Board Support package, this group interacts with the hardware and is using HAL libraries. The BSP is custom made for the specific board. The BSP shall not be using anything from the ASW.

ASW is Application Software where the high-level logic is implemented. The ASW does not directly access HAL libraries or hardware but uses BSP to allow switching to another board with the same API defined through BSP header files.

CLI (Command Line Interface) is a framework for dispatching commands from Command Line.

Drivers contains high level logic of the device driver such as packet decoding or memory management for EEPROM.

Algorithms is where the detection (Section 4.1) and the classification (Section 4.2) algorithms are.

LIB is for third-party libraries such as for BME280 and code that can be shared with BSP and ASW such as the definition of generally used types, for example, the return value `LibError_RetVal_e`.

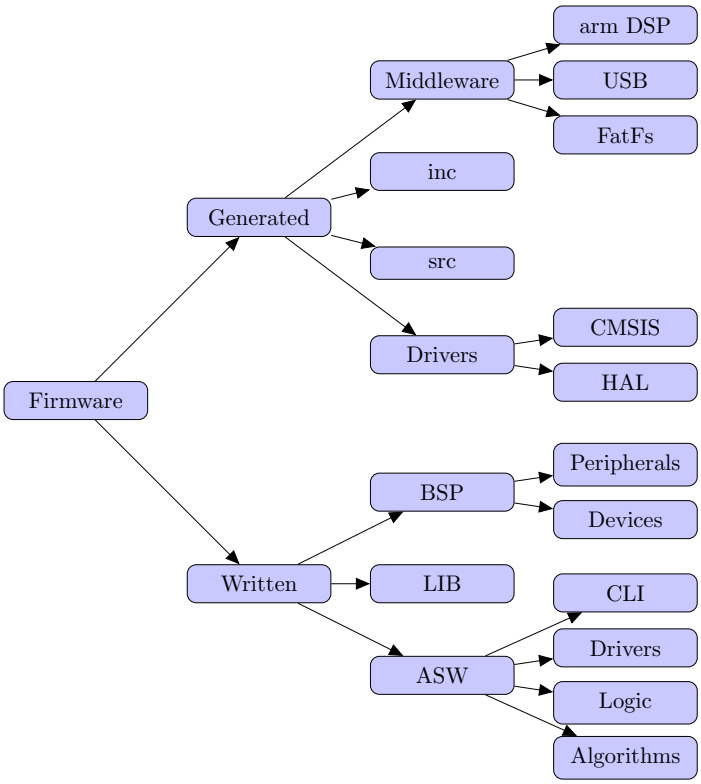


Figure 2.14: Firmware structure.

2.3.2 Interfaces

The interfaces are commonly used for the interaction of MCU with other units or with the outside world. MCU contains several interfaces like UART, I^2C, SPI,...

Universal Asynchronous Receiver-Transmitter – UART

The Universal Asynchronous Receiver-Transmitter (UART) is one of the simplest buses of all and is a full-duplex serial point-to-point communication between two peers. The physical connection uses two wires (Rx, Tx). The UART communication is block-oriented with 7 or 8 bits of data with an optionally additional one parity bit. Both peers have to have configured the same communication speed.

The used MCU can provide an interrupt signalizing the bus is idle. It enables notification about the end of a packet during packet-oriented communication with other devices. Without searching for the start and end of a packet, MCU can do other tasks. Moreover, the receiving is done via DMA, so the MCU is not loaded with handling incoming communication.

Inter-Integrated Circuit – I^2C

Interface Inter-Integrated Circuit (I^2C) is a synchronous, multi-slave serial communication bus. It is used for the connection of low-speed peripherals to MCUs in a short distance. The physical connection is via two wires (SDA and SCL). In our case, MCU is used as a master and other devices with the appropriate address defined by the manufacturer are slaves, see Table 2.1.

Device	Address	Bus	Enum
BME280	0x76	2	`E_BSP_I2C_MAX`
MAX17260	0x36	1	`E_BSP_I2C_BOSH`
M24C16 page 0	0x50	1	`E_BSP_I2C_MEM0`
M24C16 page 1	0x51	1	`E_BSP_I2C_MEM1`
M24C16 page 2	0x52	1	`E_BSP_I2C_MEM2`
M24C16 page 3	0x53	1	`E_BSP_I2C_MEM3`
M24C16 page 4	0x54	1	`E_BSP_I2C_MEM4`
M24C16 page 5	0x55	1	`E_BSP_I2C_MEM5`
M24C16 page 6	0x56	1	`E_BSP_I2C_MEM6`
M24C16 page 7	0x57	1	`E_BSP_I2C_MEM7`

Table 2.1: I^2C device address table.

For communication with devices over I^{2}C a C module was created. The module needs for communication with the desired slave a device `enum` identifier only and handles operations between the device and a higher level of the program. All the needed information such as the slave's address and bus are stored inside the module.

```
static const SlaveInfo_t m_atI2cDevs[E_BSP_I2C_CNT] = {
    /* Bus,                 Addr,             Device */
    {E_BSP_I2C_BUS_I2C2,  0x76u},          /* BOSH */
    {E_BSP_I2C_BUS_I2C1,  0x50u},          /* MEM0 */
    {E_BSP_I2C_BUS_I2C1,  0x51u},          /* MEM1 */
    {E_BSP_I2C_BUS_I2C1,  0x52u},          /* MEM2 */
    {E_BSP_I2C_BUS_I2C1,  0x53u},          /* MEM3 */
};
```

Listing 2.3: I^{2}C slave definition sample source code.

Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) is a synchronous full-duplex serial communication primarily used in embedded systems for short-distance communication. SPI is a four-wire bus with individual slave select (SS, CS) lines for multiple slave devices, master output (MOSI), slave output (MISO), and the clock signal (CLK).

The module created for handling communication between SPI devices and the program allows two modes of communication: blocking and non-blocking (DMA) transfer. The selection of the appropriate device is based on `enum` identifier.

```
/* SPI devices table */
static const DevInfo_t m_atSpiDevs[E_BSP_SPI_CNT] = {
    /* Port,        Pin,        Bus,              Device */
    { ADC_CS_Port, ADC_CS_Pin, E_BSP_SPI_BUS_SPI1 } /*
        E_BSP_SPI_ADC */
};

/* SPI bus states */
static BusInfo_t m_aeSpiStatus[E_BSP_SPI_BUS_CNT] = {
    /* Status,                  Selected,       BUS */
    {E_BSP_SPI_STATUS_READY,  E_BSP_SPI_CNT}, /* SPI1 */
};
```

Listing 2.4: SPI slave definition sample source code.

2.3.3 Command Line Interface

The command-line module is created as a communication interface between MCU and human. The command-line interface is capable of allowing password-protected access with first-level command multiplexers and help command. In the initialization of the module, it accepts array of `AswCli_Command_t` where every array entry is a top level command.

After entering the command, the input parameters are tokenized, so the signature of the functions shall mimic the known signature of the C main function on the operating system `int main(int argc, char **argv)`.

```
1  typedef struct{
2    const uint8_t        *au8CmdString;
3    const uint8_t        *au8HelpString;
4    LibError_RetVal_e (*fCallback)(const uint32_t in_u32Argc,
         const uint8_t *in_pau8Argv[]);
5  }AswCli_Command_t;
6
7  #define ASW_CLI_GENERIC_RESET_CMD {\
8    .au8CmdString = (uint8_t*)"RESET",\
9    .au8HelpString = (uint8_t*)"RESET\t-\tReset MCU",\
10   .fCallback =     AswCliGeneric_Reset,\
11 }
```

Listing 2.5: Sample defininition of CLI command.

2.3.4 Data Storing

The application has two different data storage requirements. The first is the storage of parameters that affect the behavior of individual parts of the application. Parameters occupy less space in memory but must be available during runs of the application. The parameters can be changed on the fly, for example, through CLI (Section 2.3.3).

The second requirement is the storage of recorded audio data. In this case, it may consume a large amount of space, the data can be subsequently transferred to another device, and therefore it is advisable to use a removable medium.

SD card

The SD card is primarily used for storing recorded audio of the acoustic events for possible post-processing. The written module `SoundStore` stores events into the BOOMS folder, the name of the file comprises the number of acoustic event. The quota for space on the SD card can be set, if space would be insufficient for adding a new recording, the oldest recording will be deleted. The age of the file is based on file descriptor created time which is set in the time of creating according to internal Real Time Clock (RTC).

EEPROM

The application parameters are stored in Electrically Erasable Programmable Read-Only Memory (EEPROM), which is a type of non-volatile memory.

The parameters are held inside module called `Datapool`. It provides getters and setters for parameters that are stored internally. It also provides a hard-coded fallback configuration in case that EEPROM is unavailable.

2.3.5 LoRa Messages

The structures are packed with the compiler's attribute `__attribute__((__packed__))` and then sent via LoRaWAN without other changes. Usually, the fields inside a struct are aligned with the system memory layout, in this case, the `uint8_t` occupies 4 B (32 bit) because of 32 bit processor. However, with the attribute, the compiler squeezes the size of fields just to the type specific size (`uint8_t` $\mapsto$ 8 bit).

Housekeeping

The housekeeping message serves as a notification that the device is alive with additional information about its current state.

Sending of this message is triggered via RTC alarm in configured time.

The housekeeping message contains position information, i.e. longitude, latitude, and altitude, and battery percentage.

There is a prerequisite that the node's position is stationary so that position sent for example once per day is more than sufficient.

```
1 typedef struct __attribute__((__packed__)){
2     double dLongitude;
3     double dLatitude;
4     double dAltitude;
5     int8_t i8Battery;  /* negative value -> unavailable */
6 }AswComm_Housekeeping_t;
```

Listing 2.6: Definition of Housekeeping packet structure.

Event

The event packet can be triggered by more events. The main reason is that an acoustic event was detected (Section 4.1) and classified as a gunshot (boom event type) by classification algorithm (Section 4.2). Other reasons can be that the box with the tamper switch is opened (alert event type), or that the battery is low.

The event packet contains `eType` that says if it is boom (gunshot) or alert type, with `eSubType` specifying the event more closely, depending on type. If the `eType` is a boom type, then the `eSubType` contains result of the classification algorithm (Section 4.2), on the other hand if the `eType` is an alert type then the `eSubType` specifies what kind of alert such as lid opened, low battery.

The other fields represent variable information that is needed for performing the localization task described in Section 4.3.

```
1 typedef struct __attribute__((__packed__)){
2     uint8_t  eType;
3     uint8_t  eSubType;
4     uint8_t  u8HumidityPercentage;
5     uint32_t u32PressurePa;
6     int32_t  i32TempMilliC;
7     uint32_t u32TimeStampSeconds;
8     uint32_t u32TimeStampNanoSeconds;
9 }AswComm_Event_t;
```

Listing 2.7: Defininition of Event packet structure.

Chapter 3

Server

The server is a computer that stores data sent from nodes, calculates the position of a gunshot based on events, and provides web administration and reporting console. For a machine to machine communication, API is available.

The server is directly accessible from the internet to enable access to the web page and LoraWAN communication with nodes. Because of this, the security needs to be taken into account.

The main programming language of the application software on server is Go [22]. The Go is an open-source statically typed programming language backed by Google. The output from the compilation process is native machine code and can be compiled for most of used operating systems like Linux, MAC OS, MS Windows, and most of UNIXes. For testing, UNIX platform FreeBSD was used.

Some of the language features are garbage collector [23], blazing-fast compilation times, and green threads [24]. The green threads (Goroutine) enable the creation of threads inside the user space instead of the kernel space. The Goroutine are managed with a runtime scheduler. This approach enables easy and efficient creation of a big number of Goroutines. It is useful, for example, for the webserver because a separate Goroutine needs to be created for every client. The runtime scheduler used with Goroutine offers cooperative scheduling with mechanisms such as channel communication, that are much simpler and effective than the scheduler for system threads.

Its syntax is influenced by C with emphasis on simplicity and safety. The language out of the box comes with a package manager, testing and performance benchmarking framework, and documentation generator with output in the form of a website.

Furthermore, the Go is used for creation of, for example, Docker [25], container system. Moreover, Uber uses Go and has open-source guidelines [26] for improving code correctness and moving runtime errors to compile-time errors, which is easier to solve.

As the main framework for http/https handling, Echo [27] from Labstack was selected because it is lightweight, high performance, and easy extensible.

3.1 Structure

The application software can be divided into the following sections (also shown in Figure 3.1):

Application contains the main application (business) logic. It operates as a "relay" between ports and algorithms, especially if an event of boom type is detected, the localization algorithm will be queued for performance.

Ports are tailored for given adapters such as database operation, queries are inside the Ports, and only functions are exposed.

Algorithms contain currently only the localization algorithm.

Adapters are external libraries for subscription to LoRaWan, performing queries, and handling connections to the database.

For a clearer code structure, the server is separated into internal packages such as `models`, `app`, `triangulate`,...

For easier data transfer between server's packages, the package called `models` was created. It contains models of data types (structures) that the server is internally using. For example `Event` structure is used in database package and API package.

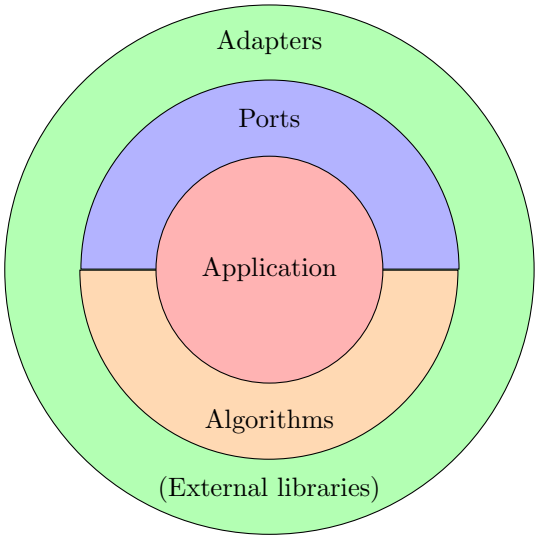


Figure 3.1: Logical structure of the application software.

3.2 Security

For the first level of security, the Transport Layer Security (TLS) is used. TLS is a cryptographic protocol designed to provide communication security, data integrity, and mainly privacy.

For managing access to the API and webpage, the so-called middleware is used. The middleware handles the connection between web server and the handler function for given Uniform Resource Locator (URL). The middleware, outlined in Figure 3.2, does the authentication of users and preprocessing of incoming requests. Authentication is a second level of security for access to API or a web page. Preprocessing/Postprocessing is needed when Gzip (compression algorithm) is used, because then outcoming messages are compressed to save data transfer and hence make faster page loading, and save the data plan on a mobile device or in case of incoming messages it performs decompression.

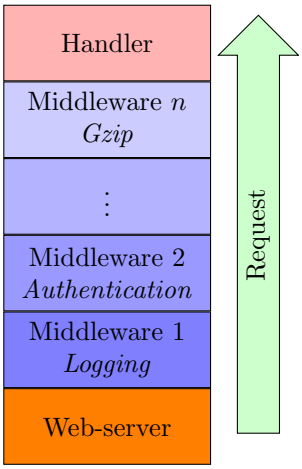


Figure 3.2: Web-server middleware layering.

3.2.1 Secure connection

The connection between client and server is always secured based on TLS enhancement. The difference between secure and unsecure connection is visible directly in the URL, when URL for secure connection is starting with https (Hyper Transfer Protocol Secure).

The process establishing TLS connection [28] contains several steps. TLS protocols use a combination of symmetric and asymmetric encryption, when the client and the server must negotiate the exchange of information about the keys and the algorithms used. The most important part of establishing a secure connection is called handshake. During the TLS handshake, the client and the server exchange all necessary information used to determine connection properties.

3.2.2 Password storing

Storing passwords in plain text without any kind of mangling is frowned upon because all database administrators can see the passwords or in case the database is hacked, the attacker can directly read the client's passwords too.

To eliminate this problem, the password salting [29] algorithm is applied for hashing of the password. In this particular case, the BlowFish cipher [30] provided in the `bcrypt` package from the standard library was utilized. The BlowFish cipher's inputs are password (thing to hash), key, and cost of the hashing. The cost of the hashing is a trade-off between security and speed.

3.2.3 Cookie management

For managing the sessions with the client, the cookie session management was selected. It means that the session information is stored on the client in the form of a cookie. Cookie has 2 string fields: name and data, because it is stored on the client who is by default untrusted, the contained information has to be secured.

The data fields are encrypted with AES-256. To improve security, the cookie has a field with time of creation to ensure that every created cookie is unique.

3.3 Database

The application needs to permanently store the data and its state. The amount of stored data can be large, so to enable fast reading and processing of data, it needs to be organised. The database offers exactly this property.

Nowadays, a large number of database systems with different properties [31] is available. The decision factor when choosing a database system is the type of the database [32] - schema-less or with schema, relational (SQL - Structured Query Language) or without relations (NoSQL).

The most known schema-less and NoSQL database is MongoDB [33], schema-less means that it does not store data inside columns but as JavaScript Object Notation (JSON), graph, etc.

The SQL database is more suitable for the data stored in this project, because the structure of the stored data is known ahead. Another reason is that the amount of data needed to store is not large, and relations for the elimination of data duplicity are convenient too.

There are many SQL databases such as MySQL [34], Microsoft SQL, PostgreSQL, MariaDB. MariaDB [35] was selected because it is an open-source alternative for already well battle-tested MySQL through industry usage. MariaDB is a drop-in replacement for MySQL.

For more user-friendly development, the DBeaver [36] community edition, which is a graphical database manager, was used. DBeaver is open-source project, which exists in a community (free) and enterprise (paid) variant.

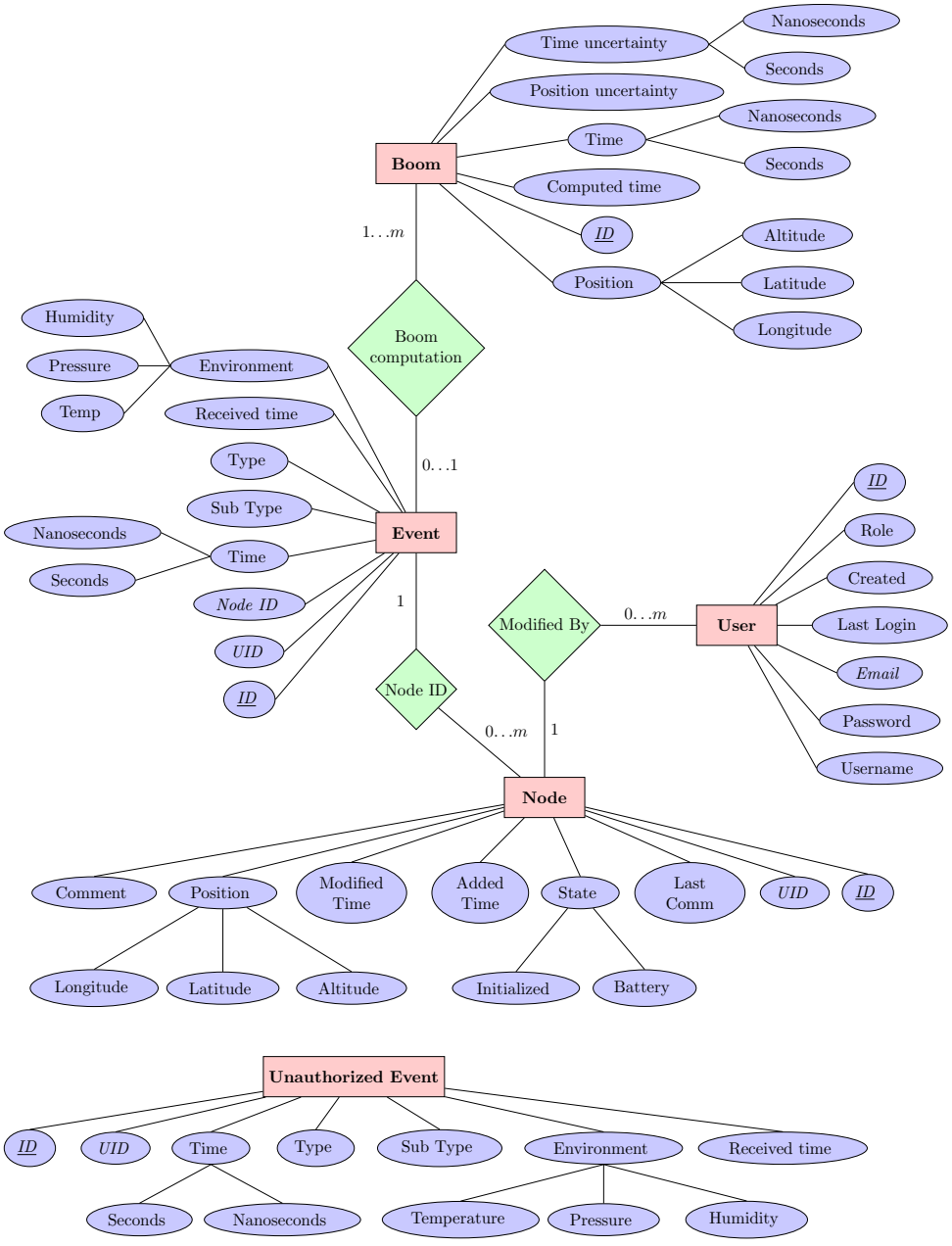


Figure 3.3: ER schema of the database.

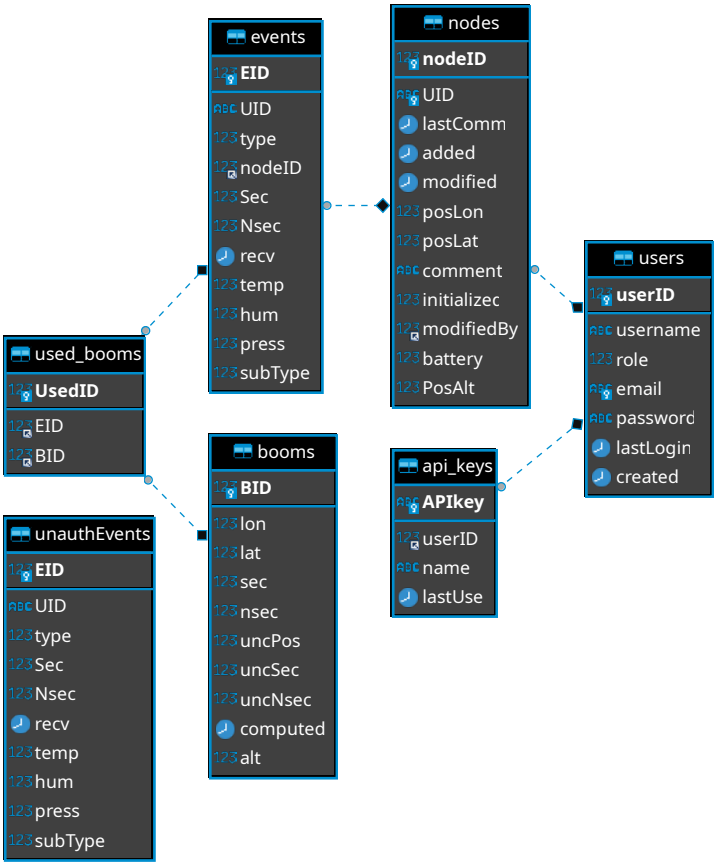


Figure 3.4: Database schema generated with DBeaver.

For the communication between application and the database, an universal `interface` modul was created. It enables easy switching between multiple database implementations. If MariaDB is for some reason insufficient, the application code does not need to be changed. Merely the driver for the MarianDB needs to be replaced by the module with new database and the `interface` remain unchanged.

At first, the ER schema of the database structure was created (Figure 3.3) and after, based on the schema, the tables and relationships were constructed (Figure 3.4).

3.4 CRUD API

The CRUD API is a machine–to–machine interface for controlling and checking the state of the application. The CRUD stands for Create, Read, Update, Delete, which means that the API supports the method for the mentioned operations. The top level structure of the CRUD API is outlined in Figure 3.5.

Other ways of API creation such as GraphQL or gRPC exist, but these interfaces are much more complicated or even need specialized clients due to the usage of technologies other than HTTP (gRPC).

The API can be accessed with two authorization models: API key or cookie session. The API key is a unique string of length 64 lowercase and uppercase alphanumeric characters that are generated by the server for the user. The user can generate the keys via the website or directly through API after logging in with the username and password.

The API is documented with Postman [37] and has 2 layers: the client layer and the admin layer.

The API call convention utilizes HTTP methods. For example, to obtain the list of all users, the HTTP method `GET` would be called on `{addr}/api/admin/getAllUsers`. When the user information needs to be updated, `PATCH` method will be called on `{addr}/api/user` URL. The creation of things is done via `POST` method.

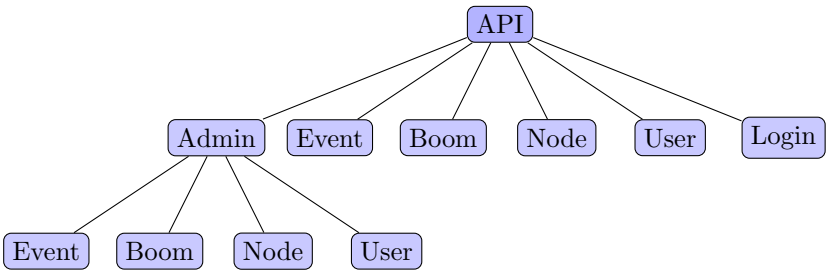


Figure 3.5: CRUD API top level structure.

All information received is validated through Go's package validator [38] from go-playground, which is able to use tags inside the definition of a struct as validation parameters.

```
1 type Pos struct {
2     Lat float64 `json:"lat" validate:"required,latitude"`
3     Lon float64 `json:"lon" validate:"required,longitude"`
4 }
```

Listing 3.1: Example of Go's structure with validation and JSON serialization/deserialization tags.

3.5 Website

The content of the website is server-side rendered, unlike modern JavaScript applications where the content is dynamically loaded and rendered on the client-side. The design of the website is in Figure 3.6.

For the rendering, Go's standard library template package is used, where tags inside the templates are replaced with data during the rendering process.

For quicker development, the plain JavaScript/HTML website template from MDBootstrap [39] was used. Inside the template, there are well-prepared web components such as layouts, dynamic tables, etc.

As maps, the Google maps were considered, but rejected, because they can not be on-premise and are closed-source. Hence, the Leaflet [40] was used. Leaflet is open-source JavaScript library for interactive maps.

If the user wants to update or insert data on the website, the JavaScript at first validates the fields and after uses the server's API described in Section 3.4 to perform the desired operation.

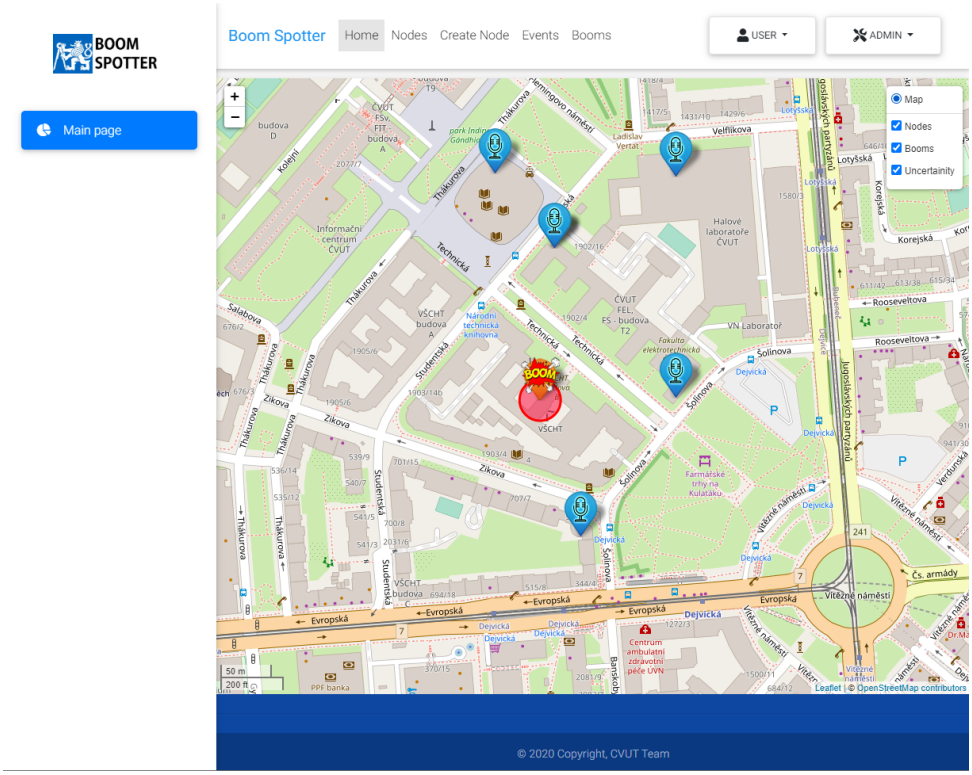


Figure 3.6: Website's home page.

3.6 Message Queuing Telemetry Transport – MQTT

The Message Queuing Telemetry Transport (MQTT) is a modern, lightweight, and compact messaging protocol for the Internet of Things (IoT). It is a publisher-subscriber type of network running over a TCP/IP stack. It is useful for the transfer of data to remote destinations, where small code size is required, mainly for M2M (machine-to-machine).

The basic features of the MQTT:

- asynchronous protocol
- compact messages
- operation under unstable transmission line conditions
- support for different levels of quality services (QoS)
- easy integration of new devices

In the MQTT protocol, messages are exchanged between a client, which can be a publisher (message provider) or subscriber (recipient of messages), and a message broker as depicted in Figure 3.7.

A publisher sends a message on a certain topic to the central point - MQTT Broker. Subscribers can receive different data from multiple publishers depending on the subscription to the corresponding topic. Every message has a topic that is a leaf of the topic's tree structure. Subscribing to a node of the tree structure means subscribing to all subsequent leaves.

In Go, `mqtt` client package for TTN [41] was utilized. It underneath uses Paho-MQTT package from eclipse [42]. The client from TTN was selected because the package has already prepared functions for subscribing to the topics.

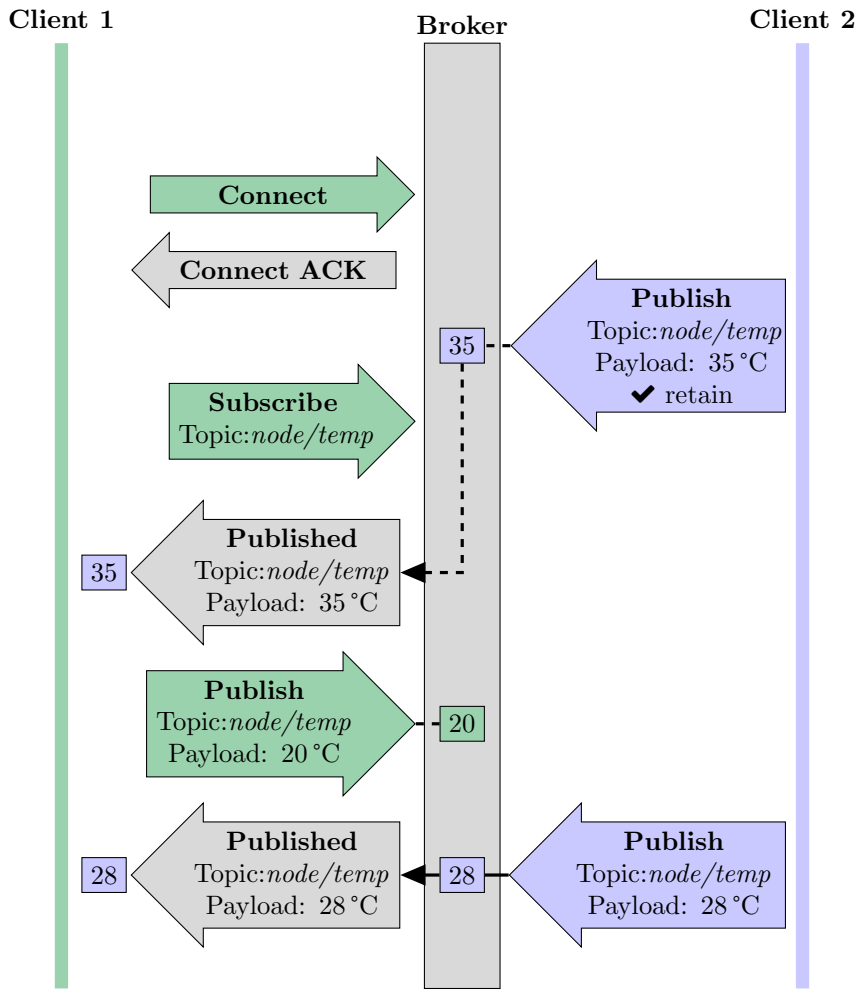


Figure 3.7: MQTT message transportation.

Chapter 4
Algorithms

The algorithms are a crucial part of the project. They provide key functionalities such as localization, acoustic event filtration, and further acoustic event classification.

Every algorithm was created and tested in Matlab for easier development and then reimplemented in the production language (C or Go).

4.1 Detection Algorithm

The peak detector (described in Section 2.1.7) is not selective enough to reliably distinguish gunshots from, for example, breaking glass. To eliminate the problem, the acoustic event detection algorithm [1] is implemented.

Execution of this algorithm is fairly cheap in terms of computing power.

The input signal is divided into taps, where the size and number of taps are hyperparameters of the algorithm. Furthermore, the algorithm has three detection parameters: level, energy, and Root Mean Square (RMS).

The median filter (described in Section 4.1.1) is used to gather an approximation of the background noise.

After the approximation of the background noise, the noise is subtracted from the middle tap which shall contain the main event (rising edge) of a potential gunshot.

From the newly created tap, the maximal value is determined and from the background noise, the RMS is calculated. These 2 values are used in the first gunshot criterion.

The second criterion is based on the median values of newly created taps around the maximal value, meaning before and after the main rising edge of the potential gunshot.

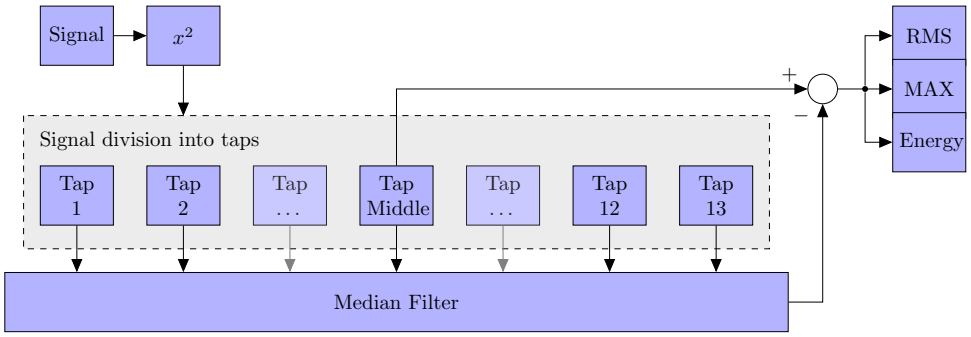


Figure 4.1: Acoustic event detection algorithm flowchart.

The algorithm is performed in the time domain (see the Figure 4.1) and consists of only simple operations such as the power of two, median filter, and root mean squared (RMS).

Parameter : $DetRMS$ = Gunshot detection RMS
$DetLevel$ = Gunshot detection level
$DetEnergy$ = Gunshot detection energy
$TapSize$ = Number of measurements in one tap
Input : S = Acoustic event signal
Result : boolean ← True if acoustic event is gunshot

begin
  $P \leftarrow S^2$;
  $Noise \leftarrow$ `MedianFilter`(P);
  $Middle \leftarrow P(middleTap) - Noise$;
  $(Val, Pos) \leftarrow$ `Max`$(Middle)$;
  if $Val > DetLevel$ **and** $Val > (DetRMS * \texttt{RMS}\,(Noise))$ **then**
    $Before \leftarrow P[Pos : Pos + TapSize]$;
    $After \leftarrow P[Pos - TapSize : Pos]$;
    $B \leftarrow$ `Median`$(Before)$;
    $A \leftarrow$ `Median`$(After)$;
    if $B >$ *(A * DetEnergy)* **then**
      return *True*
    else
      return *False*
    end
  else
    return *False*
  end
end

Algorithm 1: Acoustic event detection.

4.1.1 Median Filter

The key component of the detection algorithm is the median filter, shown in Figure 4.2. The input parameter is an array containing multiple taps (Figure 4.3) and the algorithm creates a new output array of tap size. For each index i in tap size, it calculates the median from values at index i from all taps, and the value is stored at index i in the output array.

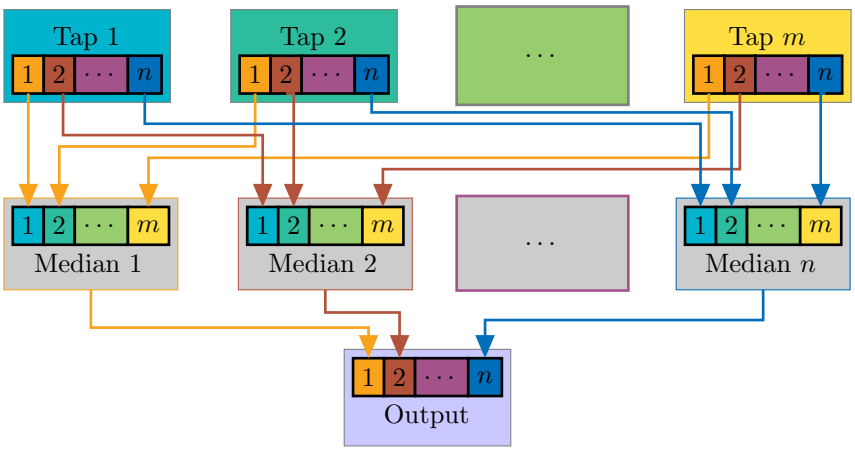


Figure 4.2: Visual median filter principle representation.

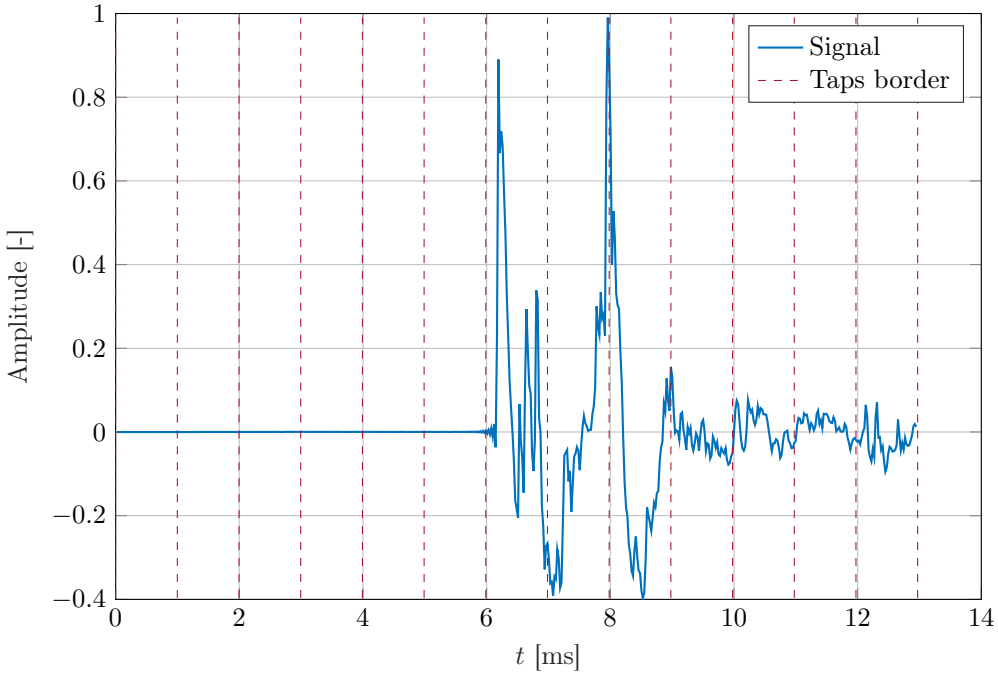


Figure 4.3: Signal divided into taps for detection.

4.1.2 Dataset

Recordings of various sounds, especially gunshots, were needed for determining the parameters of algorithms.

The available dataset [43] provided by the supervisor contains in total 247 recordings, where some recordings contain more than one acoustic event.

Unfortunately, all recordings were not created in the same place under the same conditions, especially some of the recordings of gunshots contain echoes, which can decrease the performance of the algorithm.

To acquire the annotation of a recording in the dataset necessary for machine processing, the name of the folder and the file is combined. Despite this, some parameters such as the sampling frequency were missing in some recordings.

Label	Recordings	Acoustic events
.22 LR	36	67
.22 LR subsonic	28	64
9 mm Luger	24	32
9 mm Luger subsonic	0	0
5.56 mm NATO	0	0
7.62 mm NATO	0	0
7.65 mm Browning	20	20
.45 ACP	0	0
Other caliber	0	0
Bio bubble wrap	16	18
Book slam	31	31
Bubble wrap	20	20
Hand clap	8	8
Door slam	24	24
Glass breaking	40	42
Other	0	0
Gunshots	108	183
Not gunshots	139	143
Total	247	326

Table 4.1: Contents of the provided dataset.

Matlab Object

A Matlab Object named `Recording` was implemented for easier manipulation with recordings.

The new object consists of fields:

Signal A vector consisting of the sound data.

Label An enumeration object `Labels` used for training purposes

Gun Boolean indicating if the recording contains a gunshot

Fs Sampling frequency of the recording

Location The name of the location where the recording was acquired. The sound characteristics can be similar in sound samples recorded at the same location.

Note Note for the recording such as what weapon and ammunition was used, special environmental conditions such as rain, motor noise in the background,...

Source A string containing information of the signal creation and each of the operations performed.

For example, a `Recording` is created from variable named x9mm_1_2_44100 and after separation into different acoustic events, the event source will result in x9_mm_1_2_4100_id_**N**, where **N** is the order of a given event.

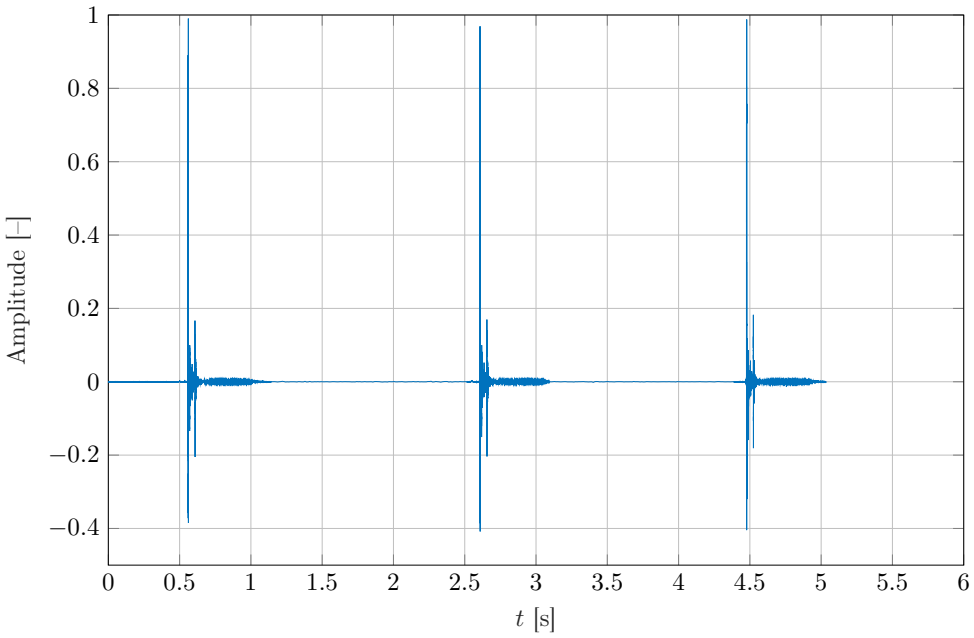
Overview of created methods:

- `AddNote`
- `GetGuns`
- `GetLabel`
- `GetLabels`
- `GetLocations`
- `GetOther`
- `GetSignal`
- `GetSignalMat`
- `GetSource`
- `HasNote`
- `IntoMfcc`
- `IntoPeaks`
- `IsFs`
- `IsGun`
- `IsLabel`
- `IsLocation`
- `IsSource`
- `Plot`
- `Recording`
- `SetGun`
- `SetLocation`
- `SetNote`

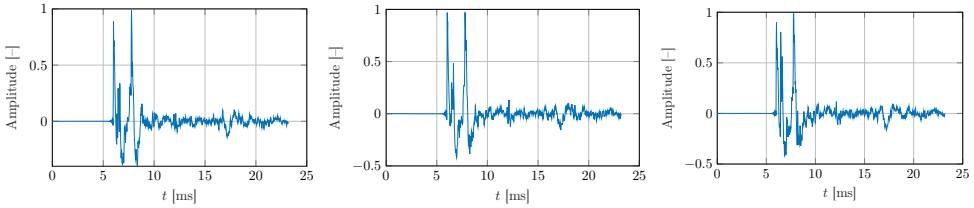
Currently, the enumeration `Labels` contains the fields listed in Table 4.1.

Method IntoPeaks. This method allows the extraction of individual acoustic events from a recording into separate standalone instances of `Recording`. An example of the extracted events is shown in Figure 4.4.

The first step of the method is finding the absolute value of the signal. Then the moving average filter is applied. This operation creates a singular peak at every acoustic event. The starting location of each peak is approximate, therefore another operation for finding the beginning of the event is used. The beginning is defined as the first value higher than the given threshold. The input signal is then cropped into separate acoustic events based on the found beginnings, the specified number of samples before the beginning of the events, and the total length.



(a) : Input signal for decomposition.



(b) : First acoustic event. **(c) :** Second acoustic event. **(d) :** Third acoustic event.

Figure 4.4: Signal decomposition into single acoustic events.

Parameter : D = Number of samples before event
L = Number of samples
T = Threshold
σ = Size of filter for finding peaks
Input : S = Acoustic signal
Result : $V \in (L, N) \leftarrow$ Acoustic events signals, where N is a number of detected events

```
1  begin
2  |  D ← |S| / Max(|S|) ;   /* Normalize and absolute signal */
3  |  idxs ← find_local_max(D, σ) ; /* Finds peaks in signal */
4  |  events ← find_init_peak(D, idxs, T) ; /* Find beginning of
   |   peaks */
5  |  i ← 0 ;
6  |  for event in events do
7  |  |  V[i, :] = S[event − D, event − D + L] ;
8  |  |  i ← i + 1 ;
9  |  end
10 end
```

Algorithm 2: Method IntoPeaks.

4.1.3 Board Implementation

Recorded signal is in `int16_t` type. In the first operation, the signal is converted into `uint32_t` with the operation power of two. This operation should not overflow because even in the worst case scenario the resulting type can contain the resulting value.

The most demanding operation on the representation is x^2, the maximal value of `int16_t` is $|-32\,768|$, power of this value is $1\,073\,741\,824$ which can be stored in `uint32_t` without any mangling.

Median computation requires sorting of the array, as a sorting algorithm `qsort()` from the standard library is used.

Testing

For testing purposes, a new module containing testing data and correct results was created. The module performs tests on these signals.

Overall, six testing signals were selected where three are guns recordings and three non-guns recordings.

Using Matlab, each selected signal is checked if the results correspond to the label. Furthermore, the parameters for the detection of these signals were set inside the testing module.

The test is successful if all gunshots are detected as gunshots and all non-gunshot signals are determined as signals without gunshot.

4.1.4 Results

The detection algorithm has to be calibrated to a given environment. When the node is permanently on one location, the parameters can be tuned to the location and therefore produce better results than in the tests conducted, where the dataset's locations differ.

The algorithm has two types of errors: False Negative (FN) and False Positive (FP). In this particular case, the number of FN shall be zero, because FN means that a gunshot happened, but the algorithm did not recognize it as such. With this precondition in mind, the parameters of the algorithm based on the provided dataset were derived as shown in Table 4.2.

The Detection Level parameter could not be determined from the provided dataset, because the parameter differs between locations and all signals had to be normalized in amplitude during preprocessing, therefore it is set to NaN.

The resulted confusion matrix is shown in Table 4.3. The correct detection is 68.20 % and the gunshots were not recognized in 2.87 % of cases.

Parameter	Value
Detection Level	NaN
Detection Energy	0.1
Detection RMS	14.5

Table 4.2: Detection algorithm parameters.

	Positive	Negative
True	169	52
False	95	5

Table 4.3: Detection algorithm confusion matrix.

4.2 Classification Algorithm

Classification of the acoustic event can potentially filter out more false-positive events that the detection algorithm (Section 4.1). It also offers additional information about possible incidents, mainly the classification of the gun caliber. For the classification, Mel-Frequency Cepstral Coefficients (MFCC) are used as the feature extractor, and afterwards the multiclass Support Vector Machine (SVM) classification is applied. The flowchart of classification is shown in Figure 4.5.

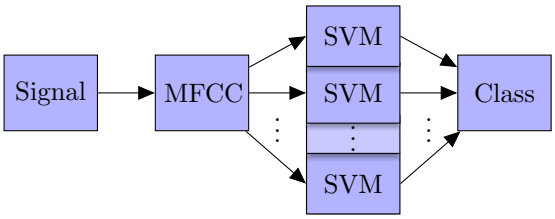


Figure 4.5: Classification flowchart.

4.2.1 Mel-Frequency Cepstral Coefficients – MFCC

Mel-Frequency Cepstral Coefficients (MFCC) are widely used in audio processing, especially for recognition like speech recording or song genre classification. The flowchart of retrieving the coefficients is in Figure 4.6

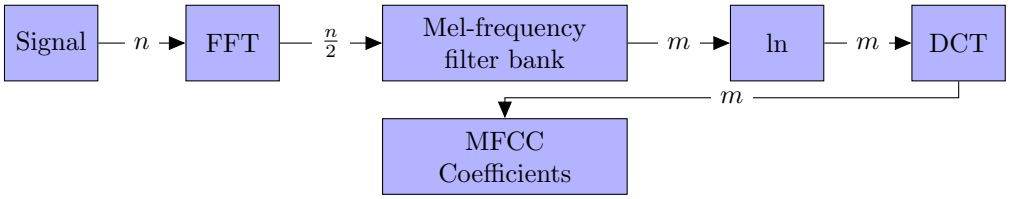


Figure 4.6: MFCC flowchart.

In the cases mentioned above, the implementation requires dividing the signal into frames, for example, a whole song cannot be taken as one frame for MFCC. The song is divided into overlapping frames and from the frames, the coefficients are extracted. In this particular case, the events are short enough that they can fit inside one frame, so framing does not need to be implemented.

For every MFC Coefficient, there is a mel-frequency filter. All mel-frequency filters create mel-frequency filter bank (Figure 4.7). The filters in the filter bank are overlaping, therefore the DCT (Section 4.2.1) is used for its decorelating properties.

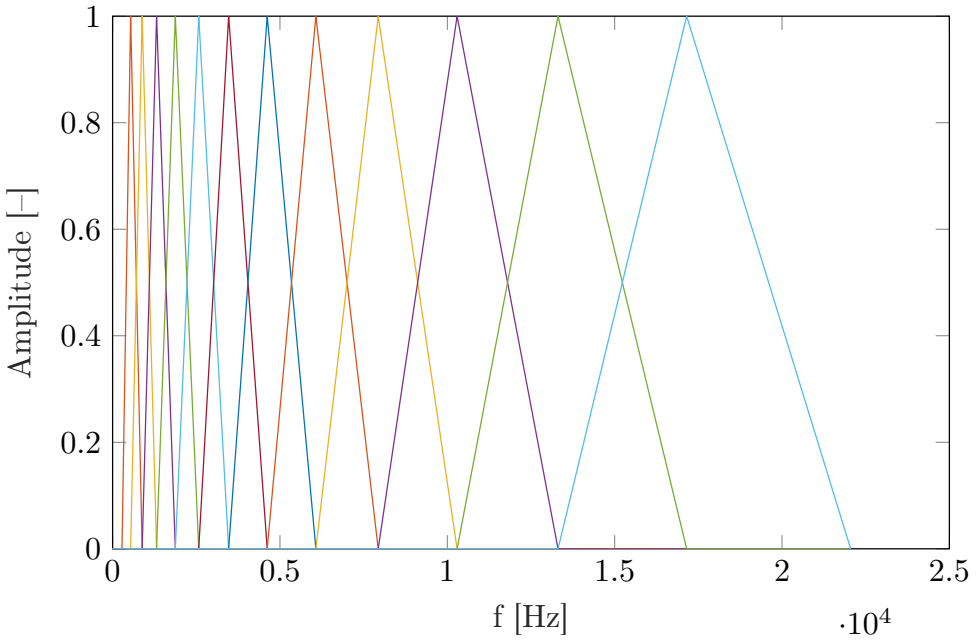


Figure 4.7: Mel-Frequency filter bank for 13 MFCC.

Mel-Frequency

Mel-Frequency (m) is one of the psychoacoustical scales. The psychoacoustical scales are defined as non-linear frequency scales which more accurately depict the change in human perception of pitch across different frequencies (f). The human ear perceives the change from 100 Hz to 200 Hz differently than 1 kHz to 1.1 kHz, although it has the same difference 100 Hz, it is generally assumed that humans perceive sound on a logarithmic scale.

Other psychoacoustical scales also exist, for example, Equivalent Rectangular Bandwidth (ERB) or Bark scale, but they are less popular than mel-frequency. The ERB scale uses a more simplified approximation of human hearing (polynomial) than the others mentioned.

The mel-frequency scale is plotted in Figure 4.8 and corresponds to the scale sensed by the human ear. The transformation functions between frequency and mel-frequency were created experimentally.

$$m(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{4.1}$$

$$f\,(m) = 700\left(10^{\frac{m}{2595}} - 1\right) \tag{4.2}$$

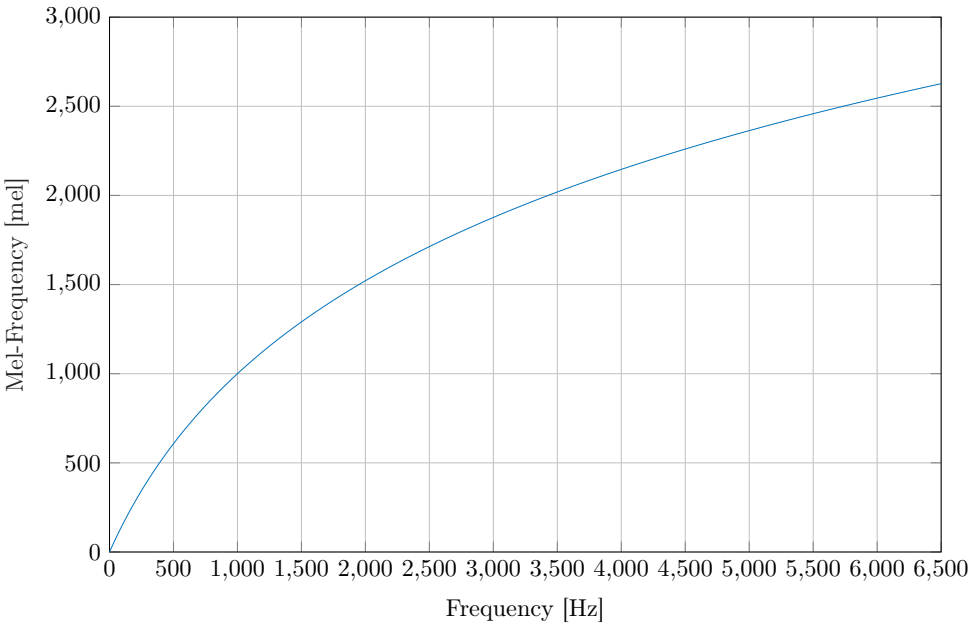
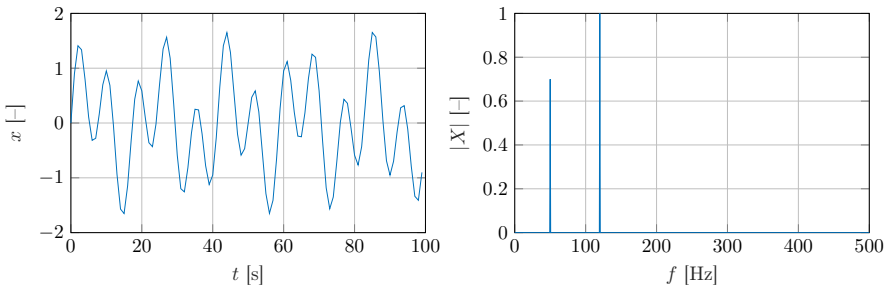


Figure 4.8: Mel-Frequency scale.

Fast Fourier Transformation

Fast Fourier transformation (FFT) algorithm computes the Discrete Fourier Transformation (DFT), which is a widely used transformation that converts a signal from the time domain into the frequency domain. The transformation can be seen in Figure 4.9. In other words, the output from the transformation is the amplitude of the sine waveforms from which the resulting signal can be assembled.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k n / N}, \quad k = 0, \ldots, N-1 \tag{4.3}$$



(a) : Time domain. **(b) :** Frequency domain.

Figure 4.9: FFT conversion from time to frequency domain, 2 sine waveforms of frequency and amplitude (50 Hz; 0.7) and (120 Hz; 1).

Discrete Cosine Transformation

The Discrete Cosine Transformation (DCT) is the same type of transformation as FFT (Section 4.2.1), but instead of sine function it uses cosines and the output is real ($\mathbb{R}^N \to \mathbb{R}^N$) instead of complex in the case of FFT.

DCT has eight variants that have a slightly modified definition, the variants are orthogonal to each other but for the scale factor.

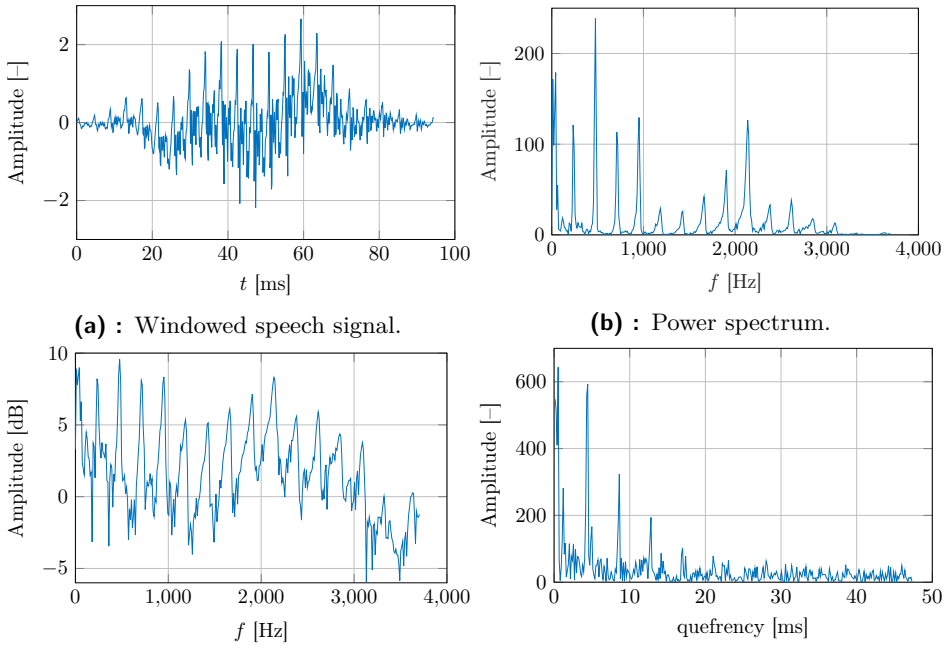
The second variant is used further with (4.4).

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} a_k x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \quad k = 0, \ldots, N-1 \\ a_k &= \begin{cases} \sqrt{\frac{1}{2}} & \text{if } k = 0 \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{4.4}$$

Cepstrum

The cepstrum is a spectrum of the logarithmic spectrum, and its creation is shown in Figure 4.10. Because of that, it can detect the periodicity inside the spectrum which can carry other information such as the rate of change in different spectrum bands.

$$C_P = \left|\mathcal{F}^{-1}\left\{\log\left(\left|\mathcal{F}\left\{x(t)\right\}\right|^2\right)\right\}\right|^2 \tag{4.5}$$



(a) : Windowed speech signal.

(b) : Power spectrum.

(c) : Log-spectrum.

(d) : Power cepstrum.

Figure 4.10: Cepstrum creation steps from a short part of audio recording of the word “Matlab”.

The cepstrum offers a new realm of definitions for lots of things in the frequency domain, in cepstrum exist alternatives such as frequency – quefrency, filtering – liftering, harmonic – rahmonic, etc. In other words the cepstrum contains quefrencies whose unit is s (second) because forward and backward DFT is used.

It can be used for canceling out echos in speech or gearbox fault detection [44]. One of the important properties of cepstrum is that the convolution transforms into addition $x_1 * x_2 \mapsto x_1 + x_2$.

4.2.2 Support Vector Machine

Support Vector Machine (SVM) [45] is a supervised learning model that is capable of regression and classification. The SVM principle is shown in Figure 4.11. In its base form, it performs linear classification with boundary maximizing the distance between different classes. In contrast to perceptron, it can also be used for linearly non-separable data by introducing penalty. SVM can also be used for non-linear separation by dimension lifting, which is a data transformation with a kernel function. There are several widely used kernel functions such as polynomial or Gaussian radial basis.

The separating hyperplane of two classes for linear classification is given by equation ($\vec{w}^T\vec{x} - b = 0$). The SVM optimal separating hyperplane is obtained by solving the quadratic optimization problem (4.6) or dual problem (4.7), where m^* is the maximal margin.

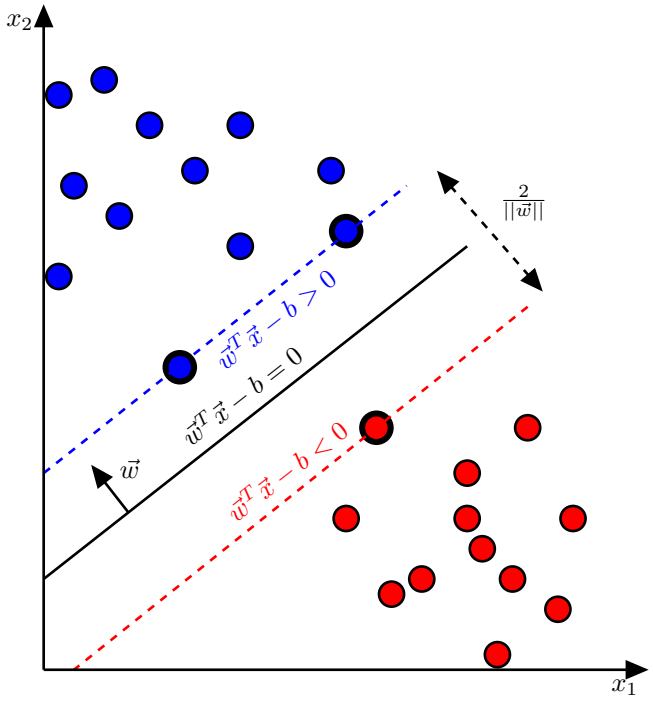


Figure 4.11: SVM principle, data points with thick black border are the support vectors.

Let $\mathcal{T}$ be a training set containing tuples of $\vec{x}_i$ point and y_i label.

$$\begin{aligned} m^* &= \max_{(\vec{w},b)} \frac{2}{\|\vec{w}\|} \\ &\text{subject to: } y\left(\vec{w}^T\vec{x}+b\right) \geq 1,\ \forall\,(\vec{x},y) \in \mathcal{T} \end{aligned} \tag{4.6}$$

$$\begin{aligned} (\vec{w}^*, b^*) &= \arg\min_{(\vec{w},b)} \left\{ \frac{1}{2}\|\vec{w}\|^2 + \sum_{i=1}^{N} f\left(\vec{x}_i, y_i, \vec{w}, b\right) \right\}, \\ \text{where} \quad f\left(\vec{x}_i, y_i, \vec{w}, b\right) &= \begin{cases} 0 & \text{if } y_i(\vec{w}^T\vec{x}_i + b) \geq 1 \\ \infty, & \text{otherwise} \end{cases} \end{aligned} \tag{4.7}$$

Multi-class Classification

The SVM can distinguish only between two classes, therefore additional methods are used, such as one-to-all or one-to-one multiclass classification.

One-to-all The one-to-all multiclass classification (Figure 4.12) works by separating one class from all remaining classes. Determining to which class a given element belongs requires in the worst case n classifications, where n is the number of classes.

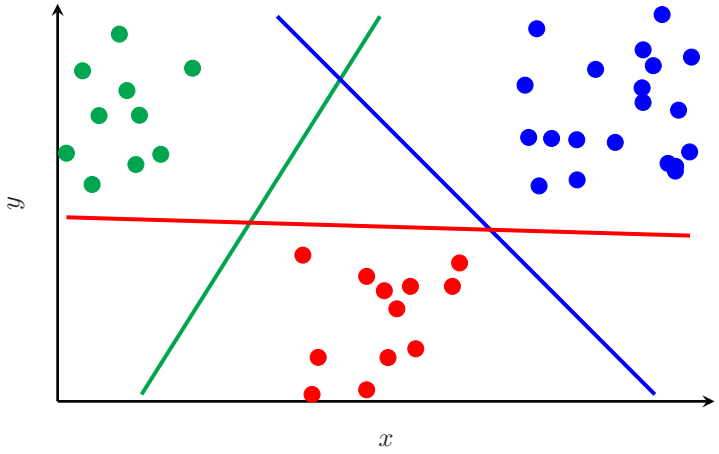


Figure 4.12: SVM one to all multiclass classification.

One-to-one One-to-one multiclass classification (Figure 4.13) takes a pair of classes and makes a classification between these two. The process is repeated for each pair. The resulting classification of a given element can be acquired as a majority of assigned classes. In the worst case, the number of classifications required is $n(n-1)$, where n is the number of classes.

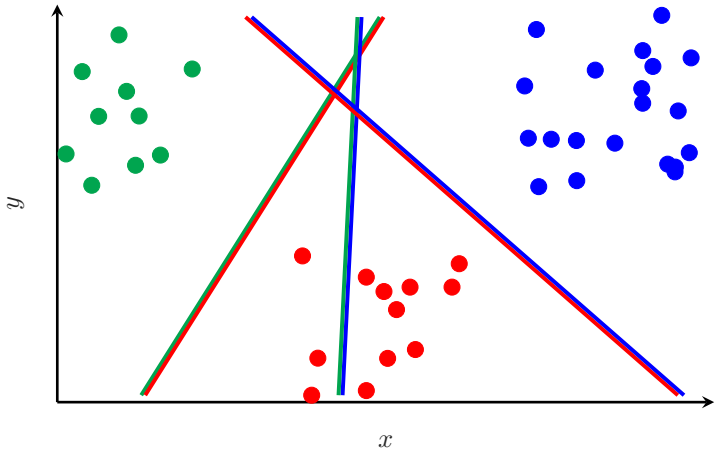


Figure 4.13: SVM one to one multiclass classification.

4.2.3 Board Implementation

In the board, a multiclass classification of one-to-all type is implemented. The current dataset (Table 4.1) includes a 5 class classification, where the fifth class is the rest (no classification), it means 4 classifications. The linear SVM with dimensionality of 13 is used, therefore the classification itself is fairly quick and simple. The first step is the classification, whether the event is a boom or not. It is conducted using a class trained on sounds of all caliber combined. If the event is classified as a boom, a narrower classification of the caliber follows.

The real implementation is that the algorithm takes a class with vector $\vec{w}$ and b and if $\vec{w}^T\vec{x} - b > 0$, the process ends with the result the class defined by $(\vec{w},\ b)$. Else the next class is selected and the process is repeated. If all classification classes were used and none selected, the result class is unknown gun caliber.

The SVMs were trained in Matlab using `fitcsvm` which returns object of SVM classifier. In the SVM object the `Beta` field is $\vec{w}$ and `Bias` field is b in case of linear kernel.

For the computation of MFCC where it is necessary to compute the FFT of the signal, the DSP [46] from CMSIS was used. The library provides a specialized FFT function for real value input, that does not need rearranging the array into complex numbers. The complex variant requires an array twice as long as the real variant because the representation of one number takes two indices.

The DCT function was implemented from the definition, see (4.4).

Mel-frequency bank starting, middle, and ending indexes of frequencies are precomputed in Matlab.

Testing

Same as in the case of Section 4.1.3 the standalone module was created just for the testing purpose of implementation of the classification algorithm.

The testing module contains five signals with their corresponding labels. Where every signal has a different class: 9 mm Luger, .22 LR, .22 LR subsonic, 7.65 mm Browning, and book slam as a not gunshot signal. The test is successful if all resulting classes of classification are the same as the reference ones.

4.2.4 Results

The whole algorithm has three parameters given by MFCC extraction: feature count, maximal frequency, and minimal frequency of mel-frequency filter bank.

Based on experience in the sound recognition industry using MFCCs [47], the number of features extracted from the signal was selected as thirteen. One of the reasons is that due to the insufficient size of the dataset, a higher number could lead to low variance and high bias (overtraining).

The minimal and maximal frequencies were found using grid search with minimizing cross-validation loss [48].

For example, the .22 LR subsonic has the highest cross-validation loss, most likely caused by the impact of the bullet in the shooting range, which is noticeable in comparison to other gunshot recordings.

Parameter	Value
Feature count	13
Maximal frequency	22 049 Hz
Minimal frequency	100 Hz

Table 4.4: Classification algorithm resulting parameters.

Class	Loss
.22 LR	0.61 %
.22 LR subsonic	11.04 %
9 mm Luger	3.68 %
7.65 mm Browning	3.07 %
Mean	4.60 %

Table 4.5: Classification algorithm error over 10-folds cross-validation.

4.3 Localization Algorithm

The localization problem is the detection of an event and its assignment to a specific location. It can be solved by several methods, namely, Time Of Arrival (TOA), Time Difference Of Arrival (TDOA), or Time Of Transmission (TOT). It is applied in different fields such as seismic (localization of the epicenter of the earthquake), radio (Mobile device localization), and surveillance (localization of gunshot).

The detection using multiple nodes is called multilateration, there are also alternatives such as trilateration which is using exactly three nodes.

4.3.1 Speed of Sound

The speed of sound c in air depends on the environmental conditions, especially on the temperature [49]. The localization algorithm uses the speed of sound for computation, so this dependency needs to be taken into account.

The used equation of the speed of sound depends on the air temperature (T [°C]), atmospheric pressure (P [Pa]) and relative humidity (H_r [%]).

$$c = \sqrt{\frac{\gamma R T_K}{M}} \tag{4.8}$$

$$T_K = T + 273.15 \tag{4.9}$$

$$P^*_{\mathrm{H_2O}} = 100 \left(1.0016 + 3.15 \times 10^{-6} \frac{P}{100} - \frac{74 \times 10^{-3}}{\frac{P}{100}} \right) \cdot 6.112 e^{\left(\frac{17.62T}{273.12T}\right)} \tag{4.10}$$

$$H = P^*_{\mathrm{H2O}} \frac{H_r}{R_w T_K} \tag{4.11}$$

$$M = M_{DA} \left(1 - H_r \frac{P^*_{\mathrm{H_2O}}}{P} \right) + H_r \frac{P^*_{\mathrm{H_2O}}}{P} m_{\mathrm{H_2O}} \tag{4.12}$$

$$\gamma = \frac{1000(1.005 + 1.82H)}{1000(1.005 + 1.82H) - \frac{R}{M}} \tag{4.13}$$

c is the resulting speed of sound in [$\mathrm{m\,s^{-1}}$].

T_K is air temperature in Kelvins.

$P^*_{\mathbf{H_2O}}$ is saturation pressure for H_2O vapors.

R is the universal gas constant (molar gas constant).

M is the molar mass of air.

H is the absolute humidity of the air.

M_{DA} is the molar mass of dry air.

R_w is the specific gas constant for water vapor.

4.3.2 Coordinate Systems

The position on the Earth can be expressed in many diverse coordinate systems. One is the spherical coordinate system using three coordinates: longitude, latitude, and altitude. Other systems are the Cartesian systems, where the system can be fixed to the rotation of the Earth (ECEF) or is inertial to the rotation (ECI), or a geocode, where the position is encoded into a set of numbers, letters, or symbols.

The ECI coordinate system is used in the Global Navigation Satellite System (GNSS) to eliminate non-Newton forces introduced by not using an inertial coordinate system, visualized in Figure 4.14.

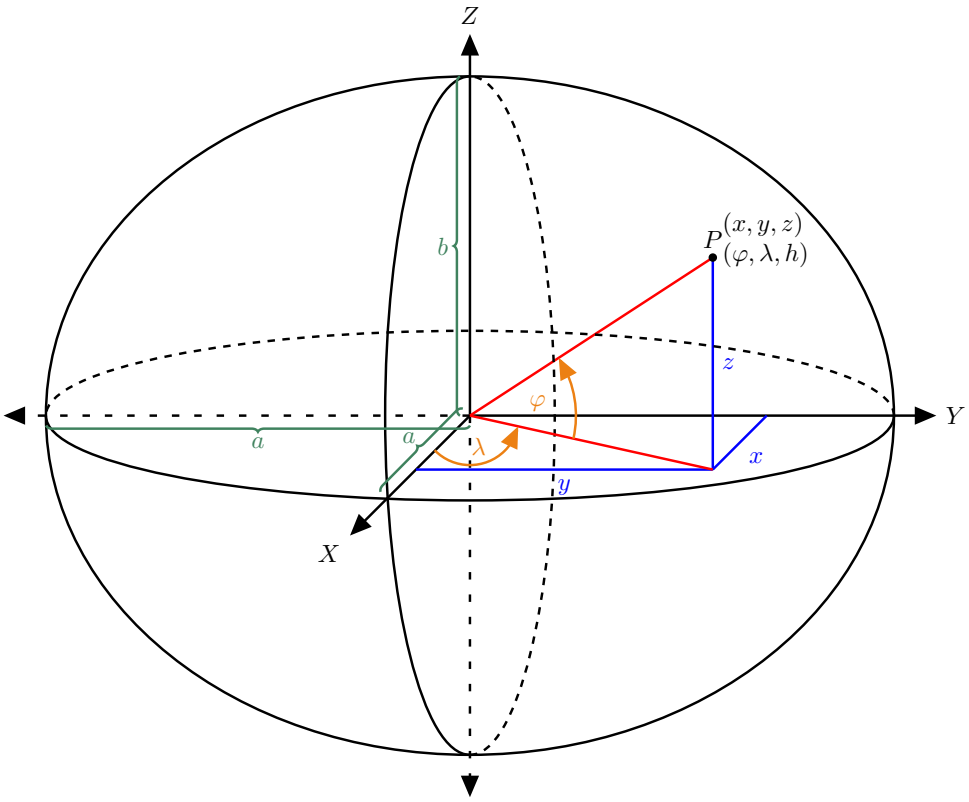


Figure 4.14: Coordinate systems, (x, y, z) ECEF, $(\varphi, \lambda, h = 0\,\mathrm{m})$ LLA.

Longitude, Latitude, Altitude

Longitude, latitude, and altitude (LLA) [50] is one of the most used position coordinate systems. It expresses the position via spherical coordinates in degrees.

The longitude (λ) defines the east-west position, where 0° is on the prime meridian, which is a line connecting both poles going through Royal Observatory, Greenwich. The value range is from −180° up to 180°.

The latitude (φ) defines north-south position, where 0° is on the Equator. The value range is from −90° up to 90° from South to North Pole.

The altitude (h) is declared as the height above the geoid (Earth's shape mathematical representation). There exists many geoids where one of the most used is WGS 84, which defines the Earth as an ellipsoid.

Earth-Centered, Earth-Fixed Coordinate System

Earth-Centered, Earth-Fixed Coordinate System (ECEF) [50] is the position in the Cartesian coordinate system with x, y and z value, where position $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ is in the center of the Earth.

There is a need for conversion from LLA into ECEF, which can be performed with (4.15), (4.16) and (4.17).

$$N = \frac{r}{\sqrt{1 - e^2 \cdot \sin(\varphi)^2}} \tag{4.14}$$

$$x = (N + h)\cos(\varphi)\cos(\lambda) \tag{4.15}$$

$$y = (N + h)\cos(\varphi)\sin(\lambda) \tag{4.16}$$

$$z = \left[\left(1 - e^2\right)N + h\right]\sin(\varphi)^2 \tag{4.17}$$

e is the Earth's model eccentricit,
in case of WGS 84 the value is $8.181\,919\,084\,262\,2 \times 10^{-2}$.

r is the Earth's model radius, in case of WGS 84 the value is 6 378 137 m.

The inverse conversion is not as much straightforward as LLA to ECEF. There are two types of conversion formulas [50] closed one, and with iteration for φ and h. The closed formula for conversion can be written as follows:

$$p = \sqrt{x^2 + y^2} \tag{4.18}$$

$$\theta = \arctan\left(\frac{z \cdot r}{p \cdot b}\right) \tag{4.19}$$

$$\lambda = \arctan\left(\frac{y}{x}\right) \tag{4.20}$$

$$\varphi = \arctan\left(\frac{z + e'^2 b \sin^3(\theta)}{p - e^2 r \cos^3(\theta)}\right) \tag{4.21}$$

$$h = \frac{p}{\cos(\varphi)} - N \tag{4.22}$$

b is semi-minor axis of the Earth model for example in the case of WGS 84 the value is $6.356\,752\,314\,245\,179 \times 10^6$ m.

e' is the second eccentricity of the Earth model for example in the case of WGS 84 the value is 0.082 094 437 949 696.

4.3.3 Basic Principle of Localization

For localization it is used a technique named multilateration, which means computing the position from the time of arrival (TOA) of energy waves, in this case acoustic. This method is widely used in aircraft navigation systems, for example, Loran-C [51].

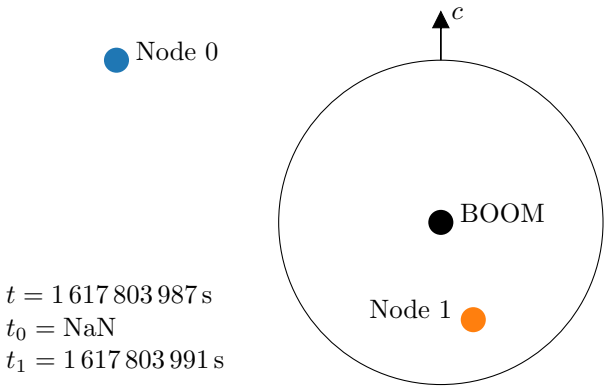


Figure 4.15: Localization problem visualization.

For the localization of gunshots, the reverse principle has to be used, meaning that the signal at time t_0 is not coming to the sensors who sense the event in times $(t_1, \ldots, t_n)$, but the sensors emit events at same times that they would receive $(t_1, \ldots, t_n)$ that in the same time (t_0) arrives at the position of interest.

When from every sensor is a wave emitted at times $t_1, \ldots, t_n$, the intersection of signals is the possible location of interest. Lines are created from the moving intersection of 2 propagating waves (circles). The resulting line is a hyperbola. The place where all hyperbolas cross is the position of interest as in the Figure 4.16.

$$r^2(t) = x^2 + y^2 \tag{4.23}$$

$$[(t_i - t)\, c]^2 = r^2 \tag{4.24}$$

$$[(t_i - t)\, c]^2 = (x - m_i)^2 + (y - n_i)^2 \tag{4.25}$$

Where $r(t)$ stands for wave propagation as in Figure 4.15 and x and y is the position on the wave at a given time t. This leads to the nonlinear system of equations, where n_i and m_i are the positions of the nodes.

For an exact number of sensors (3–4), a closed formula for this problem exists. However, the closed formulas do not account for real-life scenarios where all hyperbolas do not cross at the same points due to errors, therefore, formulas containing a type of iteration method are used, because they are capable of finding an approximate solution.

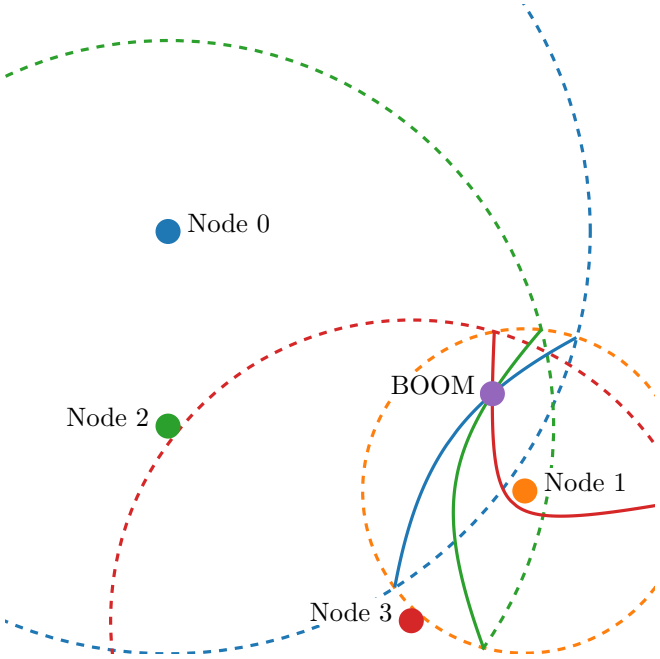


Figure 4.16: Inverse localization problem visualization.

4.3.4 3D Localization

3D localization requires at least four sensors. Less than four sensors causes the system of equations to be underdefined, because four values are calculated: position (x_0, y_0, z_0) and time of the event (t_0). This algorithm uses 5 or more sensors, because in the case of four sensors, the 2D variant (Section 4.3.5) offers better results.

$$f(\vec{r},t) = \|\vec{r}-\vec{n}\|\,\vec{\mathbb{1}} + c\,(t-t_n)\,\vec{\mathbb{1}} - \sqrt{\sum_{i=1}^{3}\left(r_i\vec{\mathbb{1}} - \vec{P}_i\right)^{②}} \tag{4.26}$$

c is the mean of all speeds of sound computed from nodes environment information.

$\vec{\mathbb{1}} \in \mathbb{R}^m$ consisting of ones.

② operator stands for element-wise power of two.

$\vec{r} = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T$, t is unknown location ($\vec{r}$) and time (t) of the event.

$\vec{n}$, t_n is the closest location ($\vec{n}$) and its recorded time (t_n).

$\mathbf{P} = \begin{bmatrix} \vec{P}_1 & \vec{P}_2 & \vec{P}_3 \end{bmatrix}^T \in \mathbb{R}^{3\times m}$ all locations that recorded event.

The 3D localization algorithm is using LM method (Section 4.3.4) for solving $\min f(\vec{r},t)$. The method is sensitive to initialization, so the initial point is taken as the localization of the node with the lowest time (it means

that it is probably the closest to the boom) with a slight shift, to prevent the loss of information while maintaining its accuracy.

The LM method also needs a derivative of the input function (4.26), in this case it is jacobian (4.27).

$$f'(\vec{r},t) = \begin{bmatrix} \frac{r_1-n_1}{\|\vec{r}-\vec{n}\|} - \frac{r_1\vec{\mathbb{1}}-\vec{P_1}}{\sqrt{\sum_{i=1}^{3}\left(r_i\vec{\mathbb{1}}-\vec{P_i}\right)^{②}}} \\ \frac{r_2-n_2}{\|\vec{r}-\vec{n}\|} - \frac{r_2\vec{\mathbb{1}}-\vec{P_2}}{\sqrt{\sum_{i=1}^{3}\left(r_i\vec{\mathbb{1}}-\vec{P_i}\right)^{②}}} \\ \frac{r_3-n_3}{\|\vec{r}-\vec{n}\|} - \frac{r_3\vec{\mathbb{1}}-\vec{P_3}}{\sqrt{\sum_{i=1}^{3}\left(r_i\vec{\mathbb{1}}-\vec{P_i}\right)^{②}}} \end{bmatrix}^T \tag{4.27}$$

Levenberg–Marquardt Method

Levenberg–Marquardt method is a nonlinear least-squares optimization method. It is working on the principle of gradient descent, but for higher stability, it introduces the regularization parameter μ.

The parameter μ is set on some initial value for example 10 000. The value is decreased if iteration of algorithm is successful (the value of criterion function lowers) for example by factor of $\frac{1}{3}$ on the other hand, if iteration is not successful the μ is increased for example with factor of $\frac{3}{2}$ and the results from the unsuccessful iteration are discarded.

In (4.29) one can see how μ changes computation of inverse matrix $\left(g'(x_k)^T g'(x_k) + \mu_k I\right)^{-1}$: if $g'(x_k)^T g'(x_k)$ is close to singular matrix the term $\mu_k I$ ensures full rank of the resulting matrix for inversion.

$$\hat{x} = \arg\min_x \|g(x)\|^2 \tag{4.28}$$

$$x_{k+1} = x_k - \left(g'(x_k)^T g'(x_k) + \mu_k I\right)^{-1} g'(x_k)^T g(x_k) \tag{4.29}$$

$$c(x) = g(x)^T g(x) \tag{4.30}$$

Parameter : t Tolerance to stop iteration
m Maximal number of iterations
μ Regularization value
Input : f Minimization function
f' Gradient of f
p Initializing point

begin
  $i \leftarrow 0$;
  $c_k \leftarrow \infty$;
  $x_k \leftarrow p$;
  while $c_k > t$ **and** $i < m$ **do**
    $i \leftarrow i + 1$;
    $x_{k+1} = \texttt{LM\_Iteration}(f, f', x_k, \mu)$; `// Iteration step (4.29)`
    $c_{k+1} \leftarrow \texttt{Criterion}(f, x_{k+1})$; `// Criterion (4.30)`
    if $c_{k+1} \leq c_k$ **then** `// Successful iteration`
      $\mu \leftarrow \texttt{Decrease}(\mu)$;
      $x_k \leftarrow x_{k+1}$;
    else `// Increase step size`
      $\mu \leftarrow \texttt{Increase}(\mu)$;
    end
    $c_k \leftarrow c_{k+1}$;
  end
end

Algorithm 3: Levenberg–Marquardt.

4.3.5 2D Localization

The 2D localization algorithm was created for event localization using only 3-4 sensors. The key components are the same as in 3D localization (Section 4.3.4), but it is operating only in two dimensions of position (x, y). This is enabled with transformation (Section 4.3.5).

There is an assumption that all sensors are at the same altitude.

The initial value of position and time is the mean value of all nodes.

$$f(\vec{r}, \mathbf{P}) = \left[c\left(r_t\vec{\mathbb{1}} - \vec{P}_t\right)\right]^{②} - \left(r_x\vec{\mathbb{1}} - \vec{P}_x\right)^{②} - \left(r_y\vec{\mathbb{1}} - \vec{P}_y\right)^{②} \tag{4.31}$$

$\vec{r} = \begin{bmatrix} r_t & r_x & r_y \end{bmatrix}$ is the resulting position and time of an event.

$\mathbf{P} = \begin{bmatrix} \vec{P}_t & \vec{P}_x & \vec{P}_y \end{bmatrix}$ is recorded time and position from all nodes.

$$f'(\vec{r}, \mathbf{P}) = 2\begin{bmatrix} -c^2\left(r_t\vec{\mathbb{1}} - \vec{P}_t\right) \\ r_x\vec{\mathbb{1}} - \vec{P}_x \\ r_y\vec{\mathbb{1}} - \vec{P}_y \end{bmatrix}^T \tag{4.32}$$

Transformation Into 2D

This transformation was created to minimize the error introduced by discarding z axis coordination from the position vector. It is made through the translation of the cluster of points to the origin (Figure 4.17) and 2 rotations around the x (Figure 4.18) and y (Figure 4.19) axes, respectively.

The transformation is made through homogeneous transformation matrices that can perform both translation and rotation in comparison to standard transformation matrices that can perform only rotation. The change is in adding one more value, that is, 1 to representation of every point $\begin{bmatrix} x & y & z \end{bmatrix}^T \rightarrow \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$.

The translation is just a subtraction of the mean position from every point's position where the homogeneous translation matrix of the vector $\vec{d}$ is defined as:

$$\mathbf{T}_{\text{trans}}(\vec{d}) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.33}$$

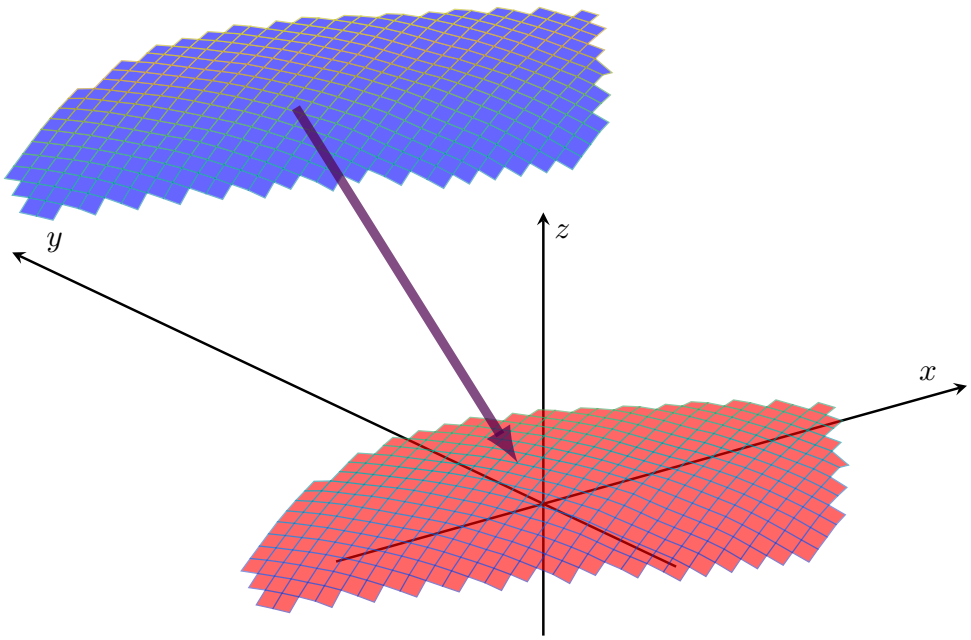


Figure 4.17: First 2D transformation operation is to move the center of mass of the points to the origin, blue before, red after.

After translation to the origin, a search for the optimal rotation can begin. The normal vector ($\vec{n}$) for the plane defined by points is acquired by eigenvalue decomposition, where the vector corresponding to the highest eigenvalue is the normal vector. At this stage, through rotation around the x and y axis, the normal vector becomes incident with the z axis.

The angle of rotation around x axis is computed as follows:

$$\varphi = \arctan\left(\frac{n_y}{n_z}\right) \tag{4.34}$$

The homogeneous rotation matrix looks like:

$$\mathbf{T}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) & 0 \\ 0 & \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.35}$$

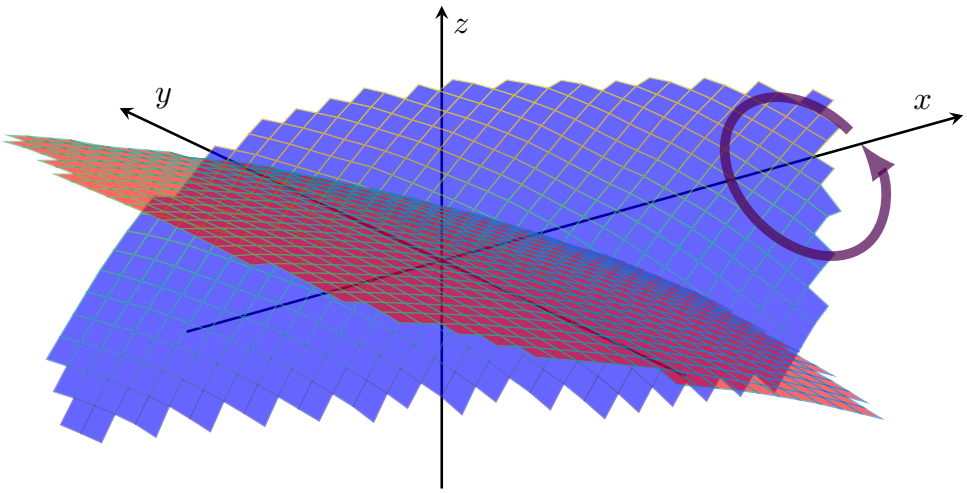


Figure 4.18: Second 2D transformation operation is to rotate around the x axis, blue before, red after.

After rotation around the x axis for computation rotation angle around the y axis there is need to transform n, because rotation around x is rotating with y. Additionally is the subtracted rotated z axis also because of x axis rotation.

$$l = \mathbf{T}_x \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{4.36}$$

$$r = \mathbf{T}_x n \tag{4.37}$$

$$\vartheta = \arctan\left(\frac{r_x}{r_z}\right) - \arctan\left(\frac{l_x}{l_z}\right) \tag{4.38}$$

The term $\arctan\left(\frac{l_x}{l_z}\right)$ returns one from two possible values π or zero based on the sign of the l_z.

The homogeneous rotation matrix that defines the second rotation around y axis is defined as follows:

$$\mathbf{T}_y(\vartheta) = \begin{bmatrix} \cos(\vartheta) & 0 & \sin(\vartheta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\vartheta) & 0 & \cos(\vartheta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.39}$$

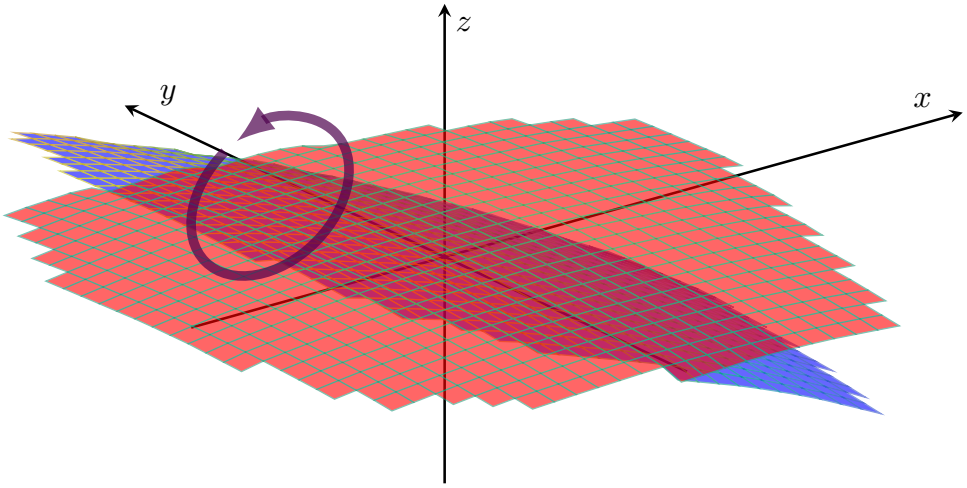


Figure 4.19: Second 2D transformation operation is to rotate around y axis, blue before, red after.

The resulting transformation matrix is calculated via multiplication.

$$\mathbf{T} = \mathbf{T}_y \mathbf{T}_x \mathbf{T}_{\text{trans}} \tag{4.40}$$

Resulting transformed positions ($\mathbf{P}$) (Figure 4.20) are computed by multiplication with transformation matrix $\mathbf{P}_{\text{new}} = \mathbf{T} \begin{bmatrix} \mathbf{P} \\ \vec{\mathbb{1}} \end{bmatrix}$.

The inverse transformation matrix is obtained by changing the sign in (v, φ, θ) to minus $\rightarrow (-v, -\varphi, -\theta)$, which is equivalent to transposition of rotation matrices and changing the sing in v to $-v$.

$$\mathbf{T}_{\text{inv}} = \mathbf{T}_{trans}(-t)\mathbf{T}_x(\varphi)^T \mathbf{T}_y(\vartheta)^T \tag{4.41}$$

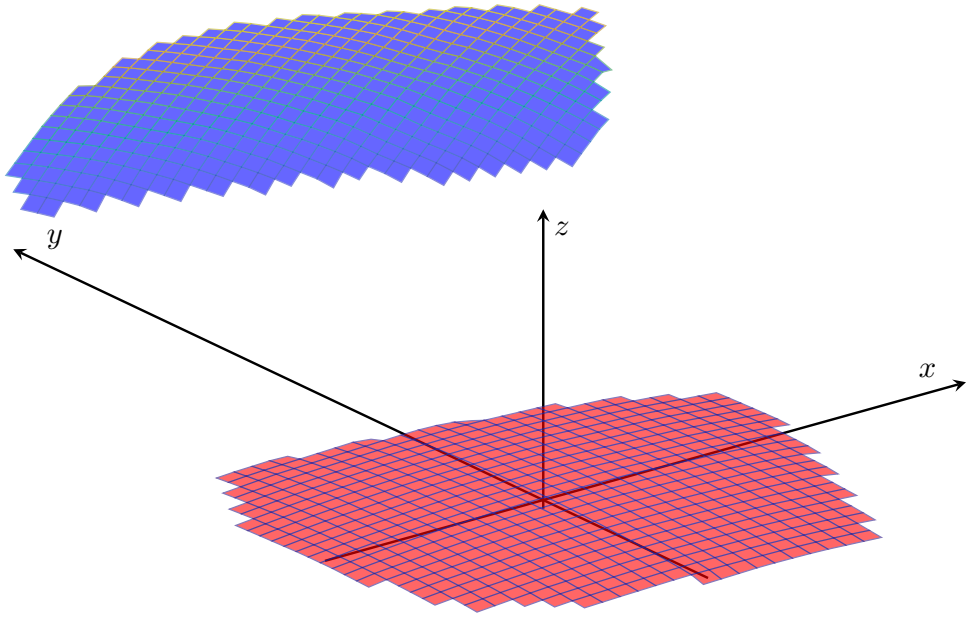


Figure 4.20: 2D transformation operations.

Input : $\Sigma \in (N, 3)$ where N is a number of points
Result : 2D Points $\in (N, 2)$

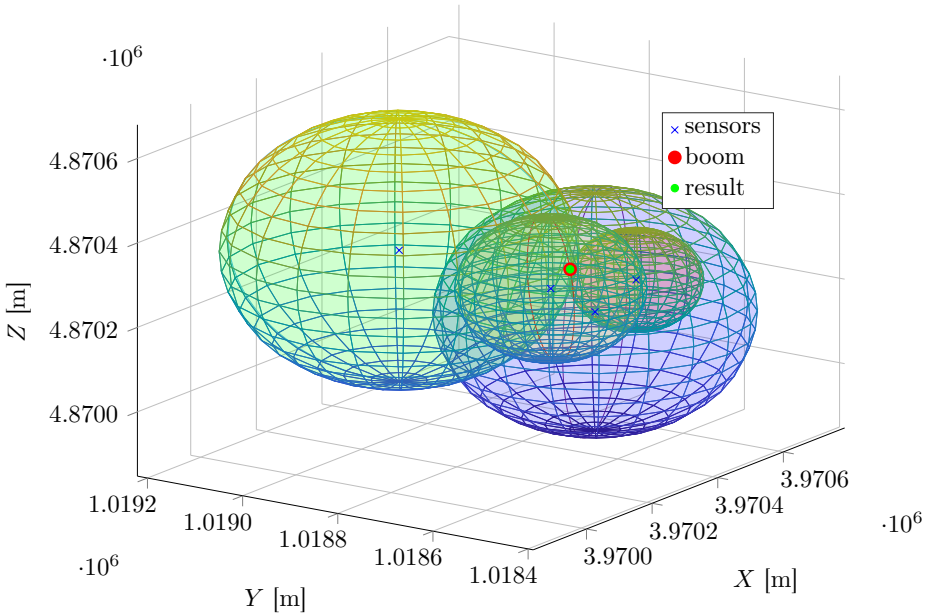
begin
  $P \leftarrow \Sigma - \overline{\Sigma}$; // Shift points to the origin
  $n \leftarrow \texttt{Eig}\ (P^T P)$; // Eigenvector corresponding to the largest eigenvalue
  $\varphi \leftarrow \texttt{atan2}(n(2), n(3))$; // Angle for rotation around x-axis
  $T_x \leftarrow \texttt{RotX}(\varphi)$; // Homogeneous Transformation Matrix for rotation around x-axis
  $n_x \leftarrow T_x \cdot \begin{bmatrix} n \\ 1 \end{bmatrix} \in \mathbb{R}^{4\times 1}$; // Transform normal vector
  $z_x \leftarrow T_x \cdot \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^T \in \mathbb{R}^{4\times 1}$; // Transform z-axis
  $\vartheta \leftarrow \texttt{atan2}(n_x(1), n_x(3)) - \texttt{atan2}(z_x(1), z_x(3))$;
  // Multiply rotation and translation homogeneous matrices
  $T \leftarrow \texttt{RotY}(\vartheta) \cdot T_x \cdot \texttt{Trans}(\overline{\Sigma})$
  $D \leftarrow \begin{bmatrix} \Sigma & \mathbb{1} \end{bmatrix} \cdot T^T \in \mathbb{R}^{N\times 4}$; // Transform Points
  return $D(:, 1:2)$
end

Algorithm 4: 2D transformation.

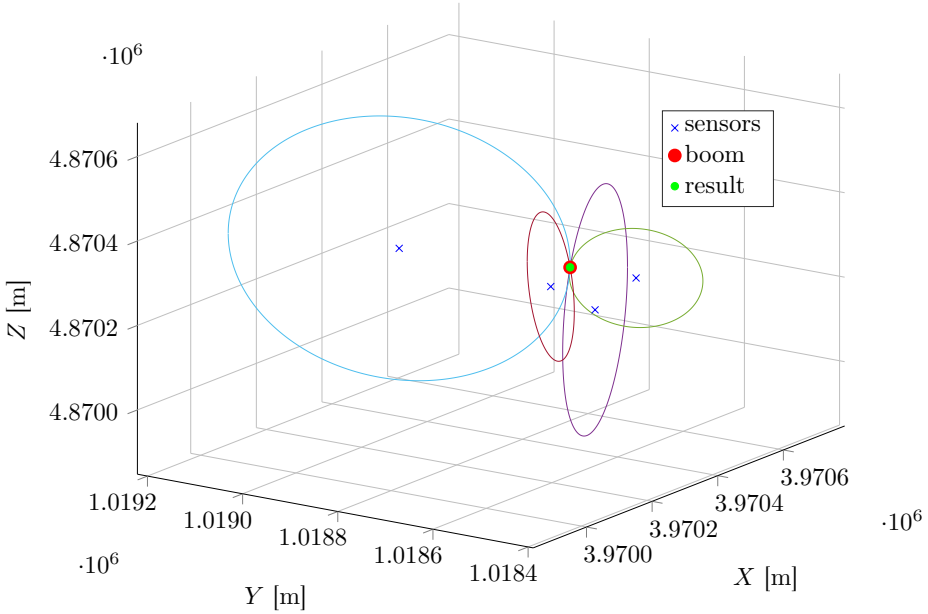
In Algorithm 4, atan2 is used instead of arctan because the atan2 function is more numerically stable and offers the same outputs.

4.3.6 Results

In a real situation, mainly in urban areas, the signal does not only propagate directly but can go through various paths caused by reflections from the walls of buildings and other structures or objects. This problem is discussed in the article [52] whose conclusion is that the multi-path propagation does not create an enormous error.



(a) : Solution with spheres.



(b) : Solution with circles going through result on the sphere for better visibility.

Figure 4.21: Localization algorithm 3D solution with 4 sensors.

The solution of localization algorithm is visualized in Figure 4.21 in 3D space. The localization algorithm sometimes results in an incorrect (non-optimal) solution. The investigation showed that the data in question had a global minimum, but also a local minimum near the global, which is not a correct solution, this behavior is visualized in 2D localization in the Figure 4.22.

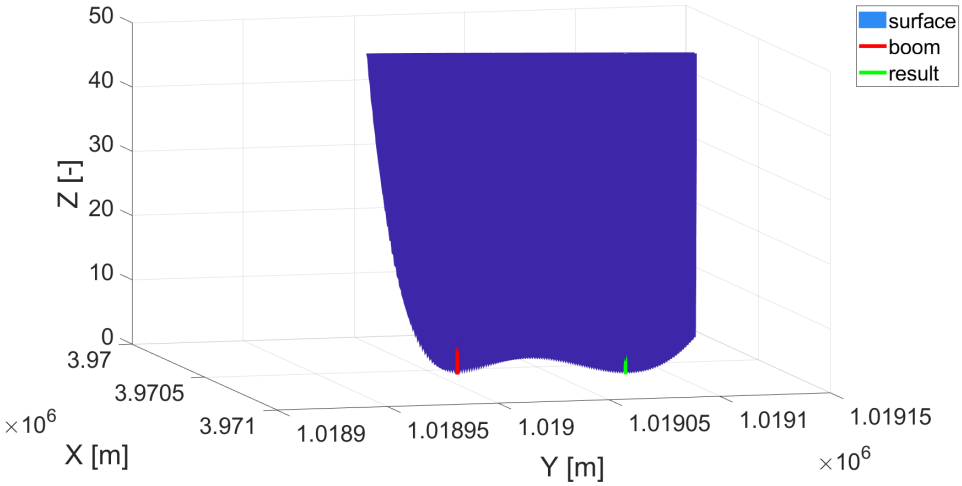


Figure 4.22: Localization problem with local and global minimum.

In case that the searched location (boom) is not inside the polygon created by nodes, the 3D localization algorithm has better results than the 2D localization.

The Figures 4.23, 4.24 were acquired with Monte Carlo method. It means that 1 000 000 locations were generated for every number of sensors and for each location and sensor, the time registered was computed with added dilatation to make the simulation closer to reality. The location of the sensors was also randomly generated. The test was performed in the geographical location of the Czech Technical University's campus in Dejvice (Table 4.6).

Corner	Longitude	Latitude
North East	50.106 213 226 093 23°	14.394 538 698 211 612°
South East	50.100 639 198 356 44°	14.395 418 462 768 497°
South West	50.100 129 933 857 18°	14.386 159 477 249 088°
North West	50.104 953 965 432 5°	14.386 717 376 724 185°

Table 4.6: Coordinates defining rectangular area from which testing location are generated.

Tests were written for the resulting production code in Go, they cover 93.3 % of the written code. The coverage is not 100 %, because some of the handling errors cannot be raised.

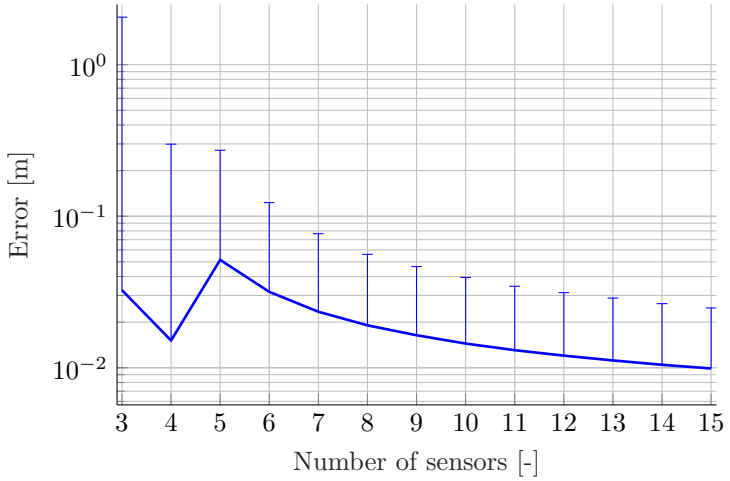


Figure 4.23: Localization mean error.

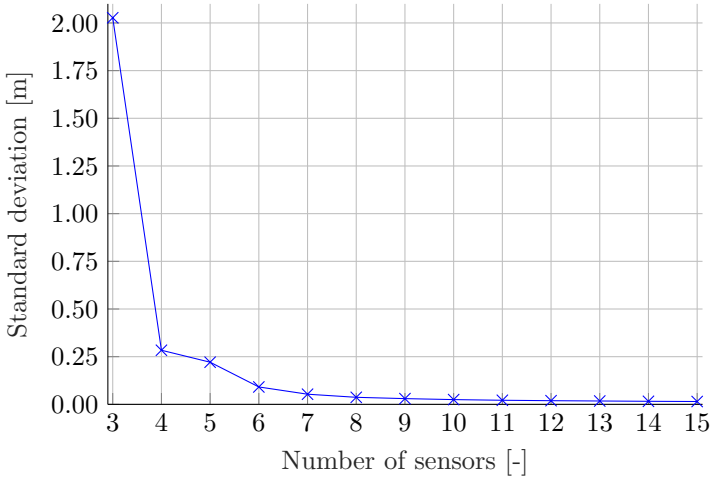


Figure 4.24: Localization standard deviation.

Chapter 5

Conclusion

The detection algorithm (Section 4.1) based on paper [1] was tested on the provided dataset [43] with successful results. Moreover, the classification algorithm (Section 4.2), which classifies the gunshot caliber with SVM classifier that uses MFCC feature extraction, showed excellent results of mean loss over classes for 10 folds cross-validation 4.6 % shown in Table 4.5, although the dataset provided for this purpose was limited.

In the implementation of the detection and especially classification algorithm, all functions except FFT were written from scratch to have complete control of properties and dependencies to assure correct reimplementation. In the case of FFT, the Matlab function and a third-party implementation (arm_math) were used and the results were compared to each other that outputs are the same.

The testing of the whole system could not be performed because of the government restrictions due to a pandemic situation. For future implementation, a real life full test should be performed.

A small test was performed with 9 mm Luger and .22 LR with short barrel where the detection and classification were successful. Audio waveform example of the recorded gunshot is plotted in Figure 5.1.

Every algorithm was tested in Matlab and after verification that everything is working correctly, reimplemented in production languages, namely C and Go. For testing of the production implementation, smaller datasets were generated with Matlab implementation.

During testing, the use of LoraWan proved to be insufficient due to its inability to send a recording of acoustic events to the server.

The Matlab tests showed that the localization algorithm (Section 4.3) is working with the error of localization at a maximum around few meters (Figure 4.23), even though the real life errors were introduced into the simulation, such as error in the precision of timestamp or error in position acquisition.

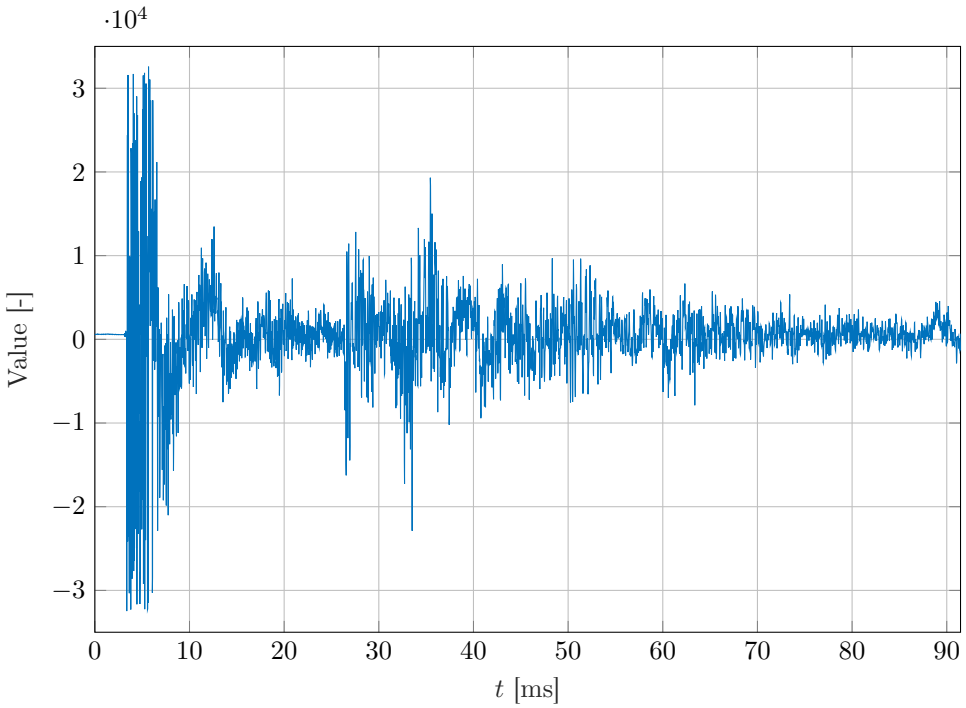


Figure 5.1: Recorded 9 mm Luger blank gunshot in closed room with the board during testing.

Bibliography

[1] Svatos, J.; Holub, J.: Smart Acoustic Sensor. *5th International Forum on Research and Technologies for Society and Industry: Innovation to Shape the Future, RTSI 2019 - Proceedings*, 2019: p. 161–165, doi: 10.1109/RTSI.2019.8895591.

[2] ShotSpotter: Home - ShotSpotter.
URL: `https://www.shotspotter.com/`

[3] V5 Systems: Wireless Gunshot Detection And Location Solution.
URL: `https://v5systems.us/products/v5psu-ptz/`

[4] Mäkinen, T.; Pertilä, P.: Shooter localization and bullet trajectory, caliber, and speed estimation based on detected firing sounds. *Applied Acoustics*, volume 71, n. 10, 2010: p. 902–913, ISSN 0003682X, doi: 10.1016/j.apacoust.2010.05.021.

[5] ST: STM32F413RH - High-performance access line, Arm Cortex-M4 core with DSP and FPU, 1,5 MByte of Flash memory, 100 MHz CPU, ART Accelerator, DFSDM - STMicroelectronics.
URL: `https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-high-performance-mcus/stm32f4-series/stm32f413-423/stm32f413rh.html#documentation`

[6] Jurasovic, M.: jurasofish/multilateration: Multilateration in 2D: IoT/LoRaWAN Mass Surveillance. 2019.
URL: `https://github.com/jurasofish/multilateration/`

[7] CRA: Služby IoT - Připojíme vaše chytrá zařízení k internetu.
URL: `https://www.cra.cz/sluzby-iot`

[8] The Things Industries: The Things Network.
URL: `https://www.thethingsnetwork.org/`

[9] ChirpStack: ChirpStack open-source LoRaWAN® Network Server.
URL: `https://www.chirpstack.io/`

[10] Home · ttn-zh/ic880a-gateway Wiki.
URL: https://github.com/ttn-zh/ic880a-gateway/wiki

[11] U-blox: u-center | u-blox. 2018.
URL: https://www.u-blox.com/en/product/u-center

[12] National, D.; Electronics, M.; Nmea, A.: National Marine Electronics Association. *History*, , n. June, 1983: p. 1–4.
URL: https://www.nmea.org/

[13] U-blox: U-Blox 6: Receiver Description Including Protocal Specification. 2011.
URL: http://www.u-blox.com/images/downloads/Product_Docs/u-blox6_ReceiverDescriptionProtocolSpec_(GPS.G6-SW-10018).pdf

[14] Maxim Integrated: ModelGauge Battery Fuel Gauge Technology - Maxim Integrated.
URL: https://www.maximintegrated.com/en/design/partners-and-technology/design-technology/modelgauge-battery-fuel-gauge-technology.html

[15] Bosch Sensortec: Bosch Sensortec BME280 sensor driver. 2020.
URL: https://github.com/BoschSensortec/BME280_driver

[16] Hakl, J.: *Jednotka pro akustickou detekci*. 2018, ISBN 9789521538070.

[17] Altium: PCB Design Software - Altium.
URL: http://www.altium.com/

[18] Onshape Inc.: Onshape | Product Development Platform. 2014.
URL: https://www.onshape.com/en/

[19] STMicroelectronics: TrueSTUDIO - A powerful eclipse-based C/C++ integrated development tool for your STM32 projects - STMicroelectronics.
URL: https://www.st.com/en/development-tools/truestudio.html

[20] van Heesch, D.: Doxygen: Doxygen.
URL: https://www.doxygen.nl/index.html

[21] ST: STM32CubeMX - STM32Cube initialization code generator - STMicroelectronics.
URL: https://www.st.com/en/development-tools/stm32cubemx.html

[22] The Go Programming Language.
URL: https://golang.org/

[23] Wilson, P. R.: Uniprocessor garbage collection techniques. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 637 LNCS, Springer Verlag, 1992, ISBN 9783540559405, ISSN 16113349, p. 1–42, doi:10.1007/bfb0017182.
URL: https://link.springer.com/chapter/10.1007/BFb0017182

[24] Toyoda, K.; Machi, K.; Ohtake, Y.; aj.: Function-Level Bottleneck Analysis of Private Proof-of-Authority Ethereum Blockchain. *IEEE Access*, volume 8, 2020: p. 141611–141621, ISSN 21693536, doi:10.1109/ACCESS.2020.3011876.

[25] Docker Inc.: Empowering App Development for Developers | Docker.
URL: https://www.docker.com/

[26] Uber Technologies Inc.: uber-go/guide: The Uber Go Style Guide.
URL: https://github.com/uber-go/guide

[27] Labstack: Echo - High performance, minimalist Go web framework.
URL: https://echo.labstack.com/

[28] Dierks, T.; Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. 2008.
URL: https://www.hjp.at/doc/rfc/rfc5246.html

[29] Gauravaram, P.: Security analysis of salt||password hashes. In *Proceedings - 2012 International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2012*, IEEE Computer Society, 2012, ISBN 9780769549590, p. 25–30, doi:10.1109/ACSAT.2012.49.

[30] Schneier, B.: Description of a new variable-length key, 64-bit block cipher (Blowfish). In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 809 LNCS, Springer Verlag, 1994, ISBN 9783540581086, ISSN 16113349, p. 191–204, doi:10.1007/3-540-58108-1_24.
URL: https://link.springer.com/chapter/10.1007/3-540-58108-1_24

[31] Cattell, R.: Scalable SQL and NoSQL data stores. *SIGMOD Record*, volume 39, n. 4, dec 2010: p. 12–27, ISSN 01635808, doi:10.1145/1978915.1978919.
URL: https://dl.acm.org/doi/10.1145/1978915.1978919

[32] Scylladb: SCYLLADB WHITE PAPER SQL to NoSQL : Architecture Differences and Considerations for Migration. 2021.

[33] MongoDB Inc.: The most popular database for modern apps | MongoDB. 2021.
URL: https://www.mongodb.com/

[34] Oracle: MySQL.
URL: https://www.mysql.com/

[35] foundation, M.: MariaDB Foundation - MariaDB.org. 2021.
URL: https://mariadb.org/

[36] DBeaver Community: DBeaver Community: Free Universal Database Tool.
URL: https://dbeaver.io/

[37] Postman Inc.: Postman | The Collaboration Platform for API Development. 2021.
URL: https://www.postman.com/

[38] go-playground/validator: :100:Go Struct and Field validation, including Cross Field, Cross Struct, Map, Slice and Array diving.
URL: https://github.com/go-playground/validator

[39] MDBootstrap.com: Material Design for Bootstrap 5 and 4 - Material Design for Bootstrap.
URL: https://mdbootstrap.com/

[40] Leaflet Development Team: Leaflet - a JavaScript library for interactive maps. 2020.
URL: https://leafletjs.com/

[41] The Things Industries: TheThingsNetwork MQTT client.
URL: https://github.com/TheThingsNetwork/ttn/tree/develop/mqtt

[42] Eclipse: Eclipse Paho MQTT Go client.
URL: https://github.com/eclipse/paho.mqtt.golang

[43] Svatos, J.; Holub, J.: Acoustic events dataset. 2019.

[44] Randall, R. B.: CEPSTRUM ANALYSIS AND GEARBOX FAULT DIAGNOSIS. *Maintenance Management International*, volume 3, n. 3, 1982: p. 183–208, ISSN 01675389.

[45] Meyer, D.; Leisch, F.; Hornik, K.: The support vector machine under test. *Neurocomputing*, volume 55, n. 1-2, sep 2003: p. 169–186, ISSN 09252312, doi:10.1016/S0925-2312(03)00431-4.

[46] Keil, R.; Krech, J.: ARM-software/CMSIS: Cortex Microcontroller Software Interface Standard.
URL: https://github.com/ARM-software/CMSIS

[47] Velardo, V.: Mel-Frequency Cepstral Coefficients Explained Easily. Technical report.
URL: https://github.com/musikalkemist/AudioSignalProcessingForML/blob/master/

19-MFCCsExplainedEasily/Mel-FrequencyCepstralCoefficientsExplainedEasily.pdf

[48] Kohavi, R.: A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Technical report, 1995.
URL: http://robotics.stanford.edu/$\sim$ronnyk

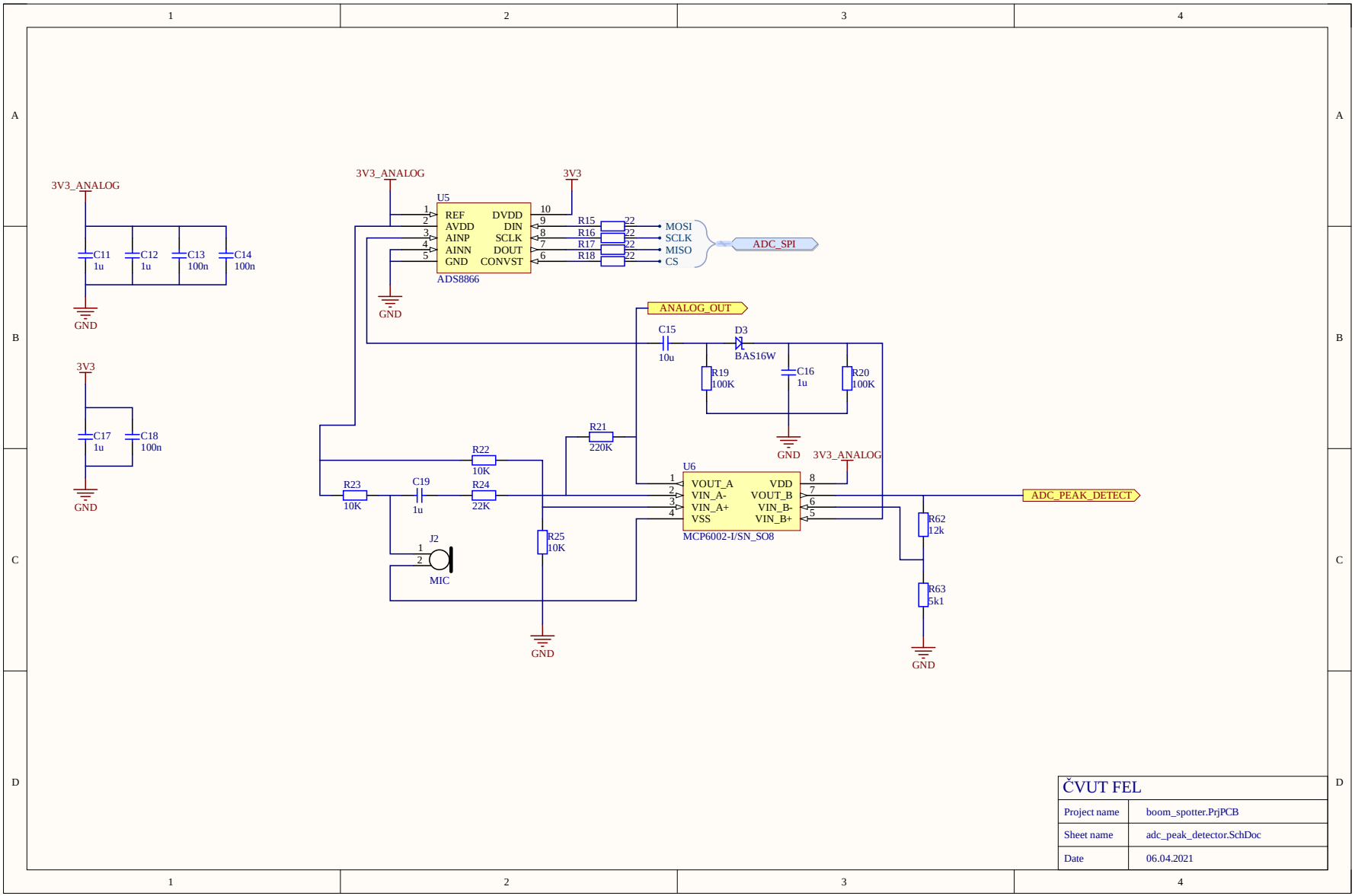
[49] Cramer, O.: The variation of the specific heat ratio and the speed of sound in air with temperature, pressure, humidity, and CO2 concentration. *Journal of the Acoustical Society of America*, volume 93, n. 5, jun 1993: p. 2510–2516, ISSN NA, doi:10.1121/1.405827.
URL: https://asa.scitation.org/doi/abs/10.1121/1.405827

[50] NAL Research: Datum Transformations of GPS Positions: Application Note. *U-blox ag*, 1999: page 12.
URL: http://www.u-blox.ch

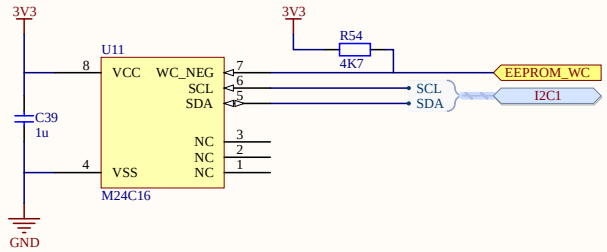
[51] Loran-C - Introduction.
URL: http://www.jproc.ca/hyperbolic/loran_c.html

[52] Holub, J.; Svatos, J.: Limits and Accuracy of Gunshot Detection. 2021.

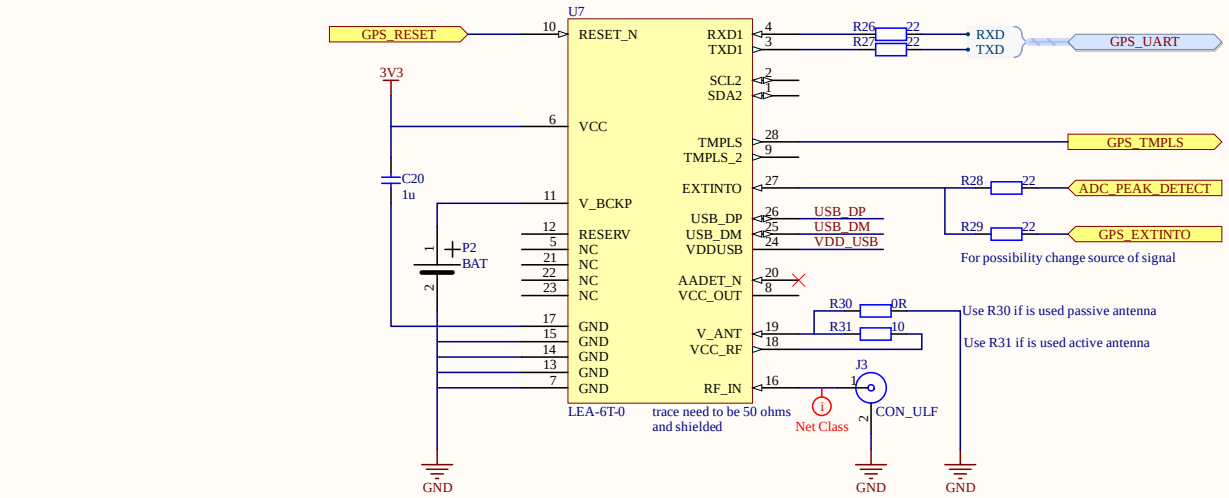
Appendix A
Board schematics

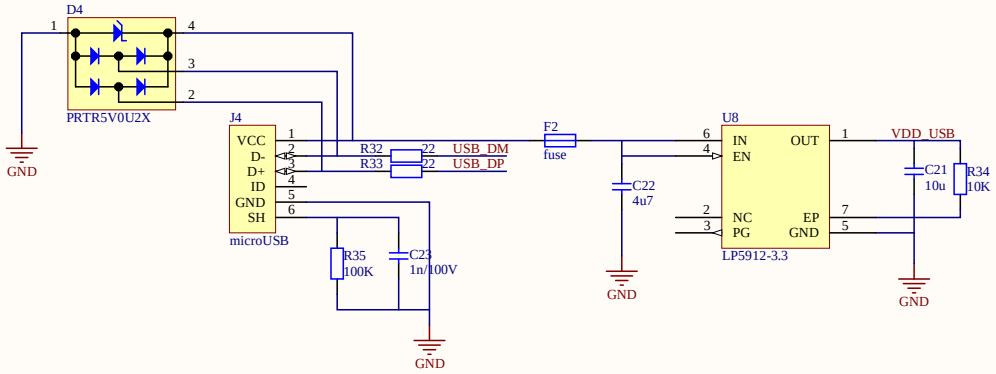


ČVUT FEL	
Project name	boom_spotter.PrjPCB
Sheet name	adc_peak_detector.SchDoc
Date	06.04.2021

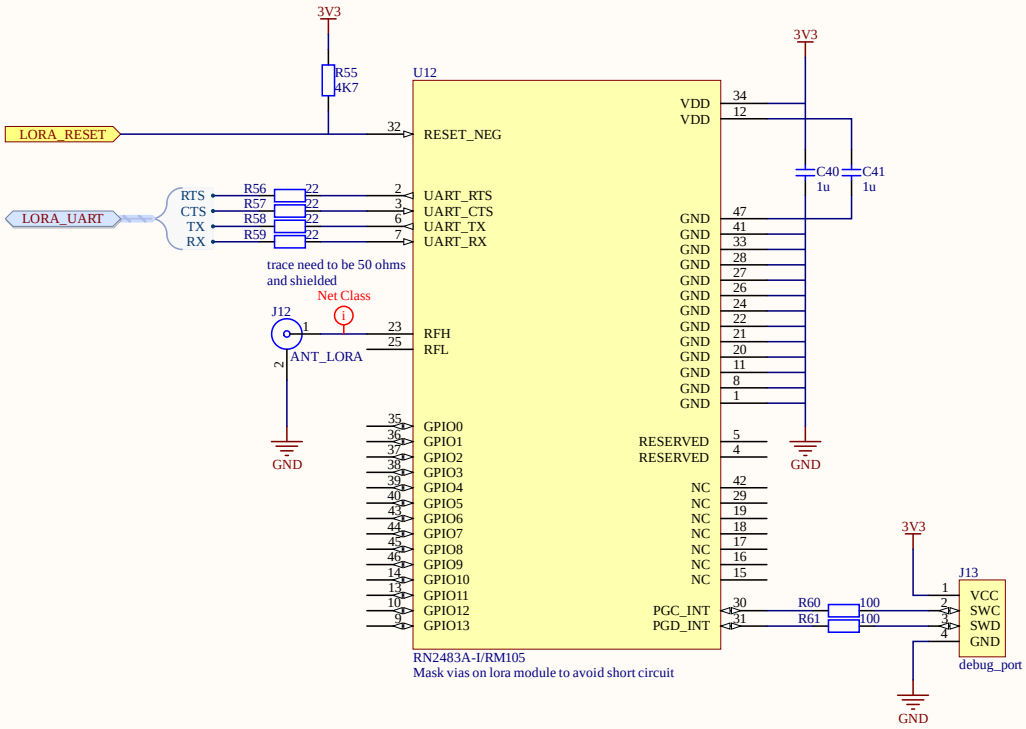


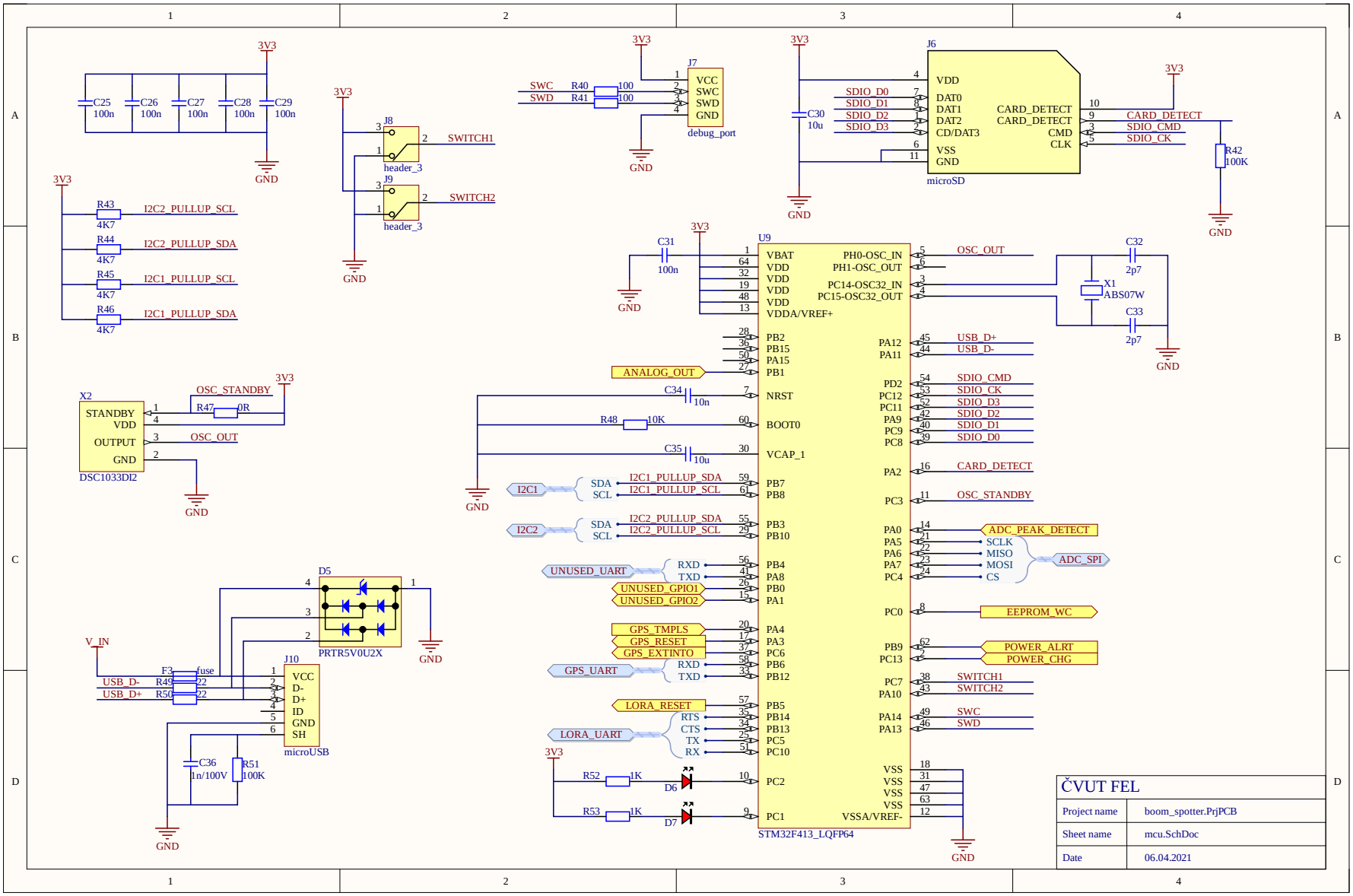
ČVUT FEL	
Project name	boom_spotter.PrjPCB
Sheet name	eeprom.SchDoc
Date	06.04.2021

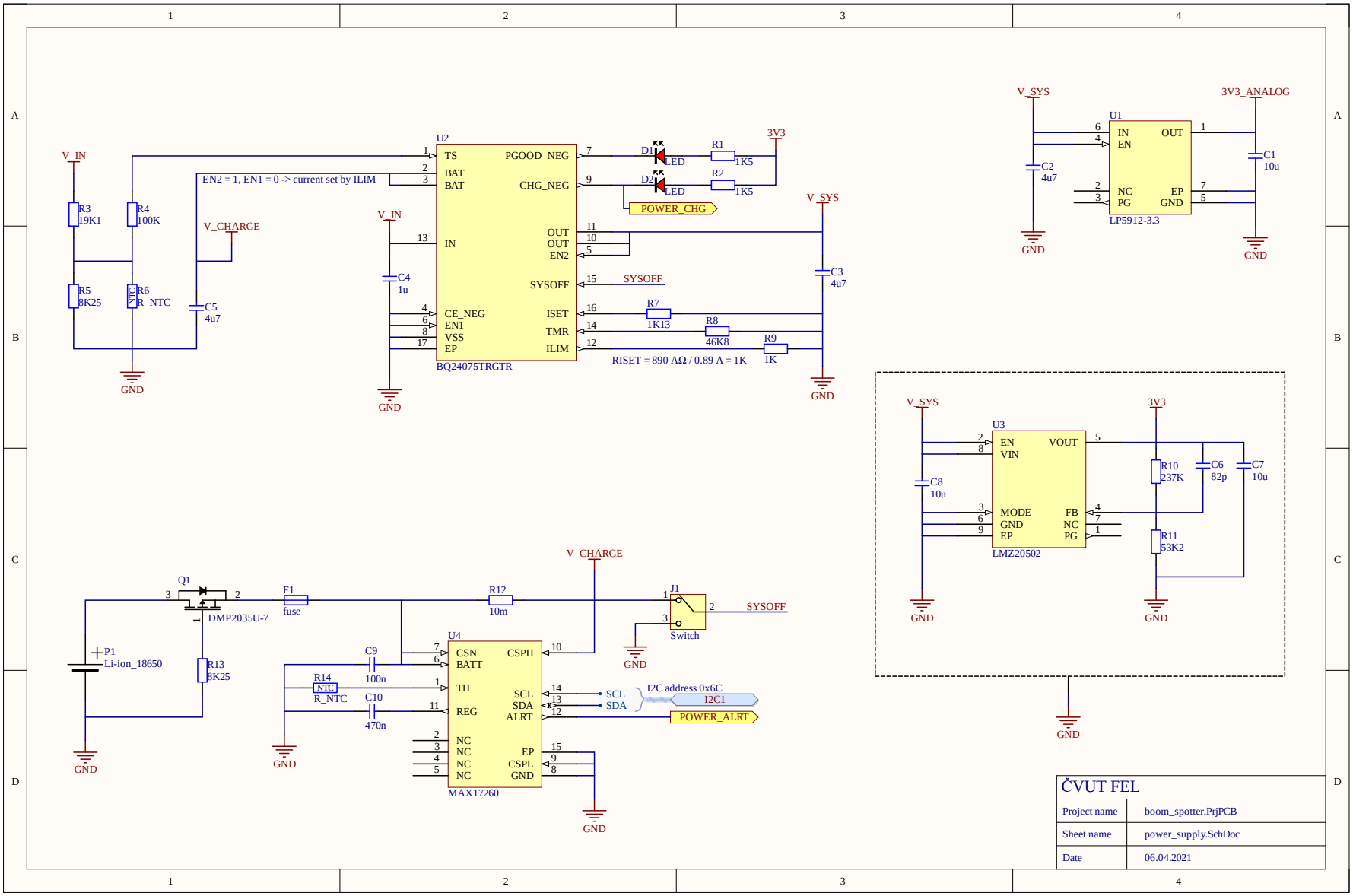




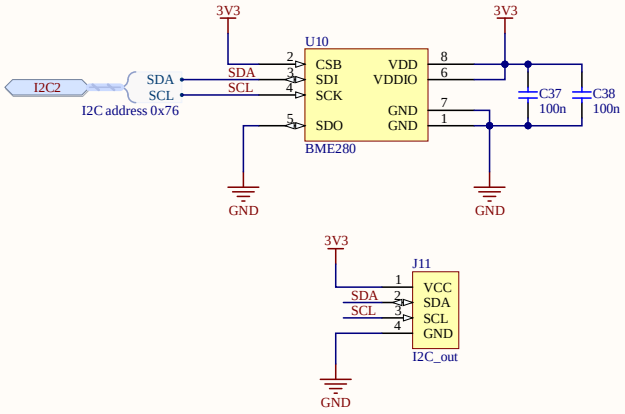
ČVUT FEL	
Project name	boom_spotter.PrjPCB
Sheet name	gps_module.SchDoc
Date	06.04.2021


3V3
R55
4K7
U12
LORA_RESET
32
RESET_NEG
LORA_UART
RTS
CTS
TX
RX
R56
R57
R58
R59
22
2
3
6
7
UART_RTS
UART_CTS
UART_TX
UART_RX
trace need to be 50 ohms
and shielded
Net Class
J12
1
2
ANT_LORA
23
25
RFH
RFL
GND
35
36
37
38
39
40
41
42
43
44
45
46
14
13
10
9
GPIO0
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
GPIO8
GPIO9
GPIO10
GPIO11
GPIO12
GPIO13
VDD
VDD
34
12
3V3
C40
1u
C41
1u
47
41
33
28
27
26
24
22
21
20
11
8
1
GND
GND
GND
GND
GND
GND
GND
GND
GND
GND
GND
GND
GND
5
4
RESERVED
RESERVED
GND
42
29
19
18
17
16
15
NC
NC
NC
NC
NC
NC
NC
30
31
PGC_INT
PGD_INT
R60
R61
100
100
3V3
J13
1
2
3
4
VCC
SWC
SWD
GND
debug_port
GND
RN2483A-I/RM105
Mask vias on lora module to avoid short circuit
ČVUT FEL
Project name
boom_spotter.PrjPCB
Sheet name
lora.SchDoc
Date
06.04.2021

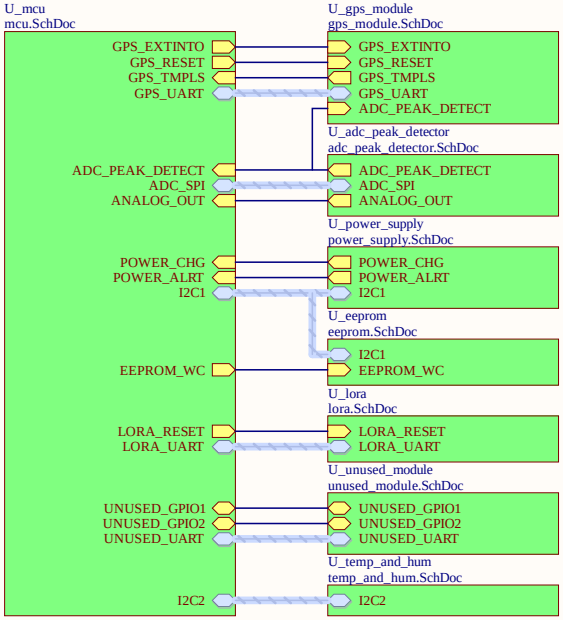

C25 100n
C26 100n
C27 100n
C28 100n
C29 100n
3V3
GND
R43 4K7
R44 4K7
R45 4K7
R46 4K7
I2C2_PULLUP_SCL
I2C2_PULLUP_SDA
I2C1_PULLUP_SCL
I2C1_PULLUP_SDA
J8
J9
header_3
SWITCH1
SWITCH2
SWC
SWD
R40 100
R41 100
J7
VCC
SWC
SWD
GND
debug_port
J6
VDD
DAT0
DAT1
DAT2
CD/DAT3
VSS
GND
SDIO_D0
SDIO_D1
SDIO_D2
SDIO_D3
CARD_DETECT
CMD
CLK
SDIO_CMD
SDIO_CK
microSD
C30 10u
R42 100K
C31 100n
U9
VBAT
VDD
VDDA/VREF+
PH0-OSC_IN
PH1-OSC_OUT
PC14-OSC32_IN
PC15-OSC32_OUT
OSC_OUT
C32 2p7
C33 2p7
X1 ABS07W
USB_D+
USB_D-
ANALOG_OUT
C34 10n
R48 10K
C35 10u
BOOT0
VCAP_1
NRST
I2C1
I2C2
SDA
SCL
OSC_STANDBY
ADC_PEAK_DETECT
ADC_SPI
SCLK
MISO
MOSI
CS
EEPROM_WC
POWER_ALRT
POWER_CHG
UNUSED_UART
UNUSED_GPIO1
UNUSED_GPIO2
GPS_TMPLS
GPS_RESET
GPS_EXTINTO
GPS_UART
LORA_RESET
LORA_UART
RXD
TXD
RTS
CTS
TX
RX
SWITCH1
SWITCH2
SWC
SWD
R52 1K
R53 1K
D6
D7
STM32F413_LQFP64
X2
STANDBY
VDD
OUTPUT
GND
DSC1033DI2
R47 0R
D5
PRTR5V0U2X
V_IN
F3 fuse
R49 22
R50 22
USB_D-
USB_D+
J10
VCC
D-
D+
ID
GND
SH
microUSB
C36 1n/100V
R51 100K
ČVUT FEL
Project name
boom_spotter.PrjPCB
Sheet name
mcu.SchDoc
Date
06.04.2021

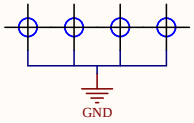


ČVUT FEL	
Project name	boom_spotter.PrjPCB
Sheet name	power_supply.SchDoc
Date	06.04.2021

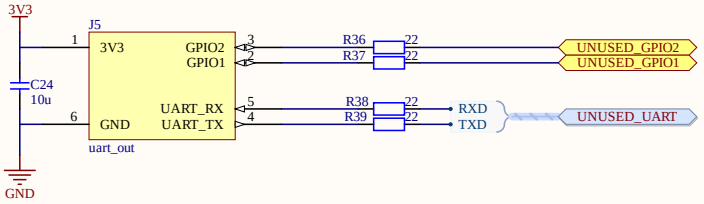


ČVUT FEL	
Project name	boom_spotter.PrjPCB
Sheet name	temp_and_hum.SchDoc
Date	06.04.2021

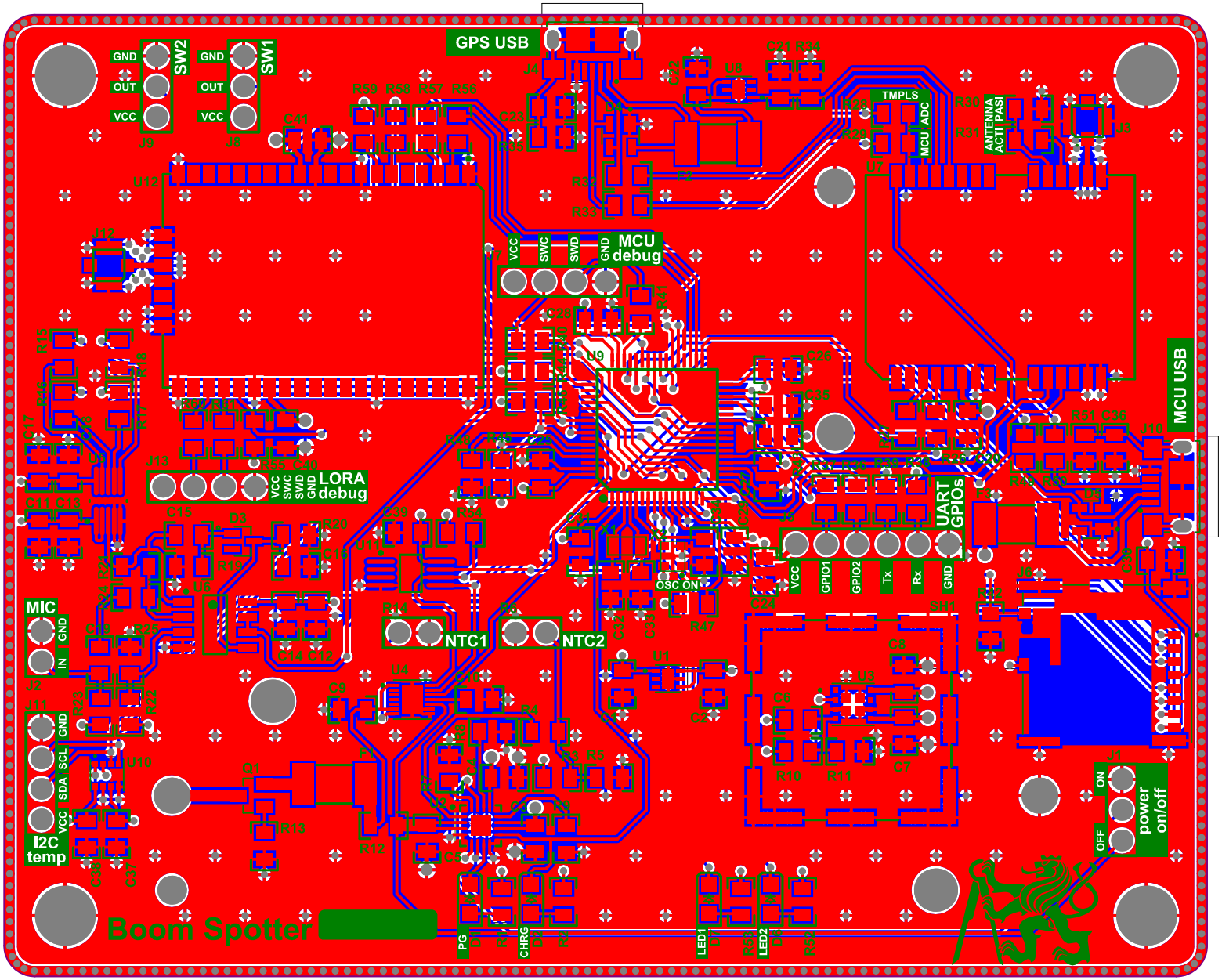


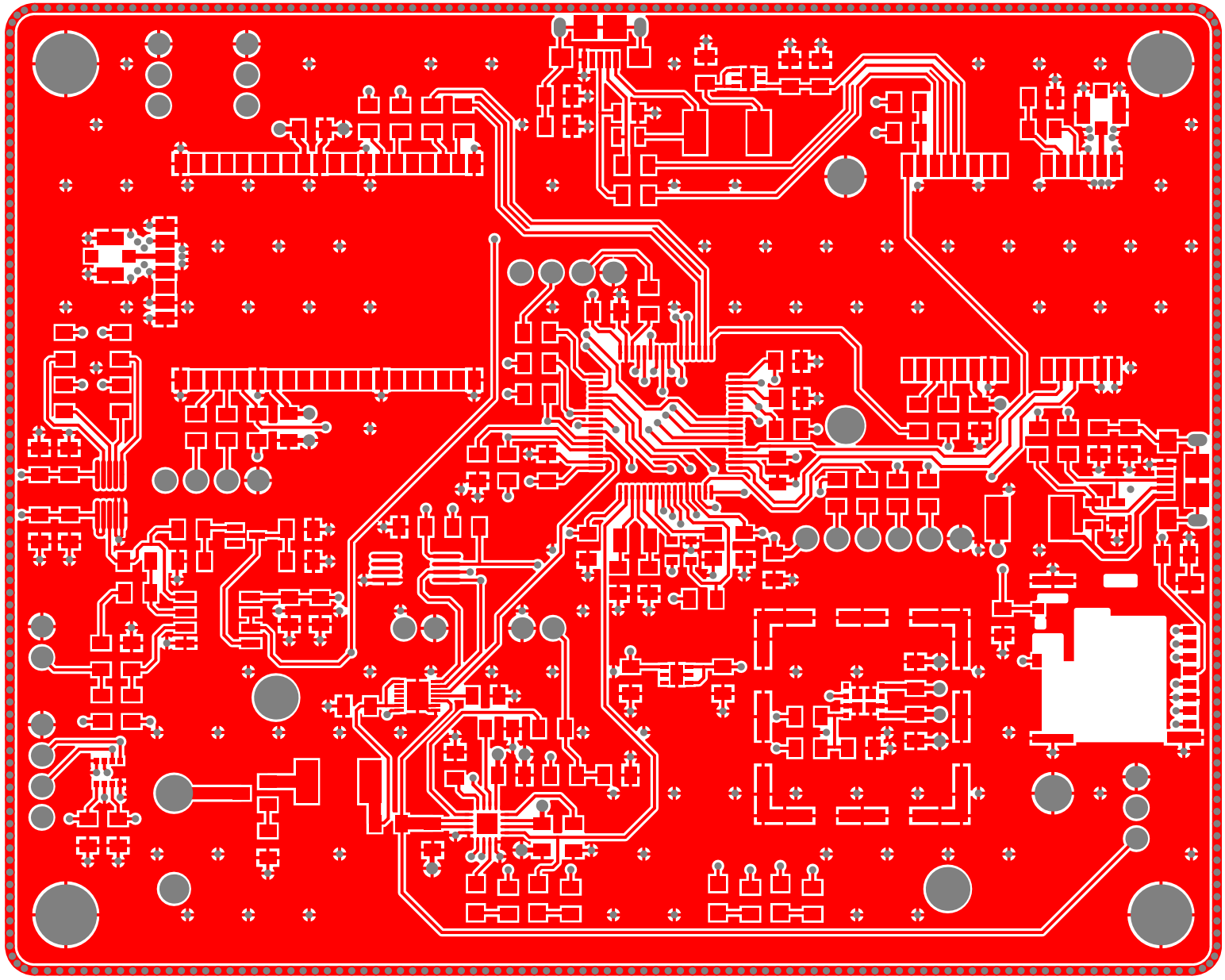


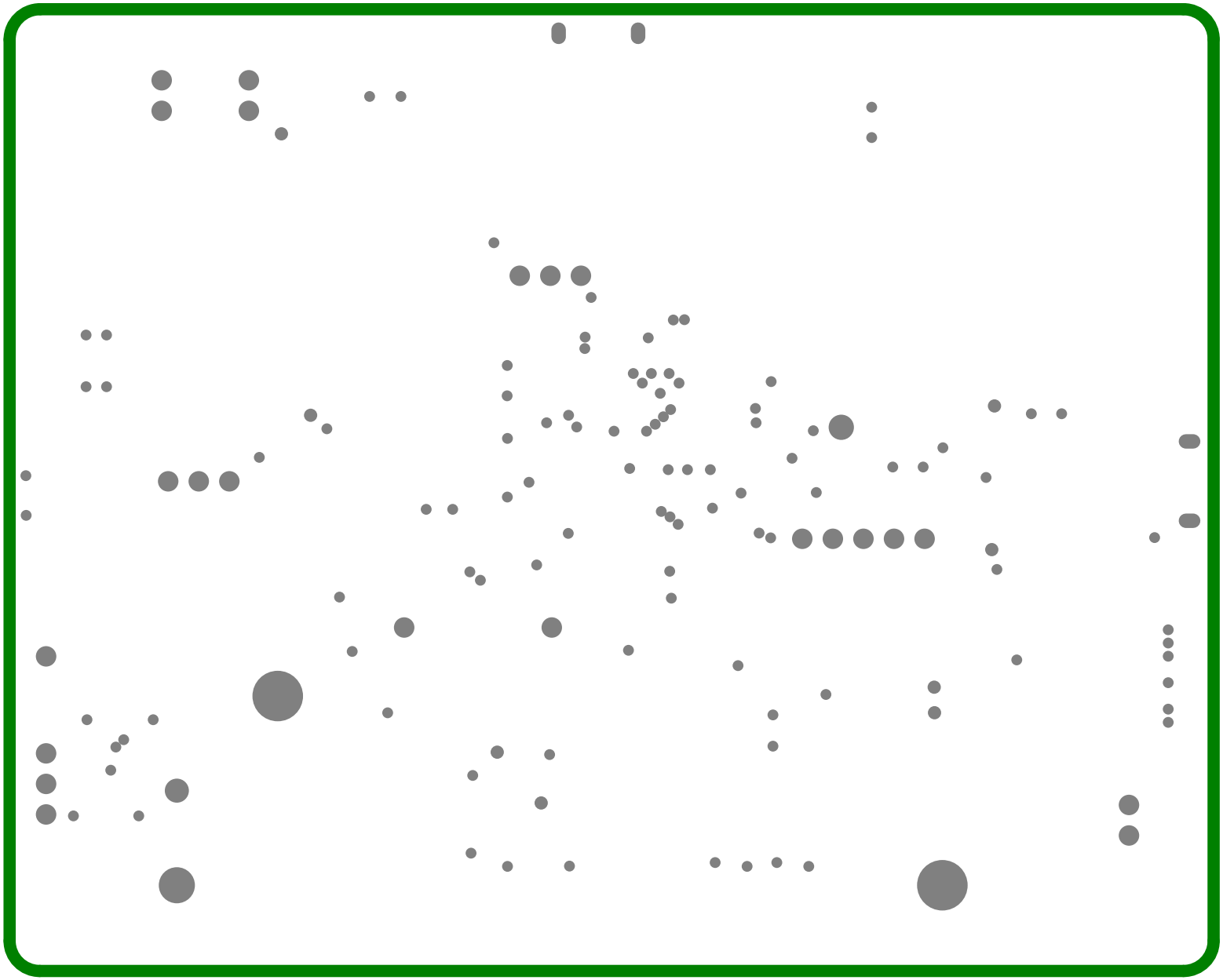
ČVUT FEL	
Project name	boom_spotter.PrjPCB
Sheet name	top.SchDoc
Date	06.04.2021

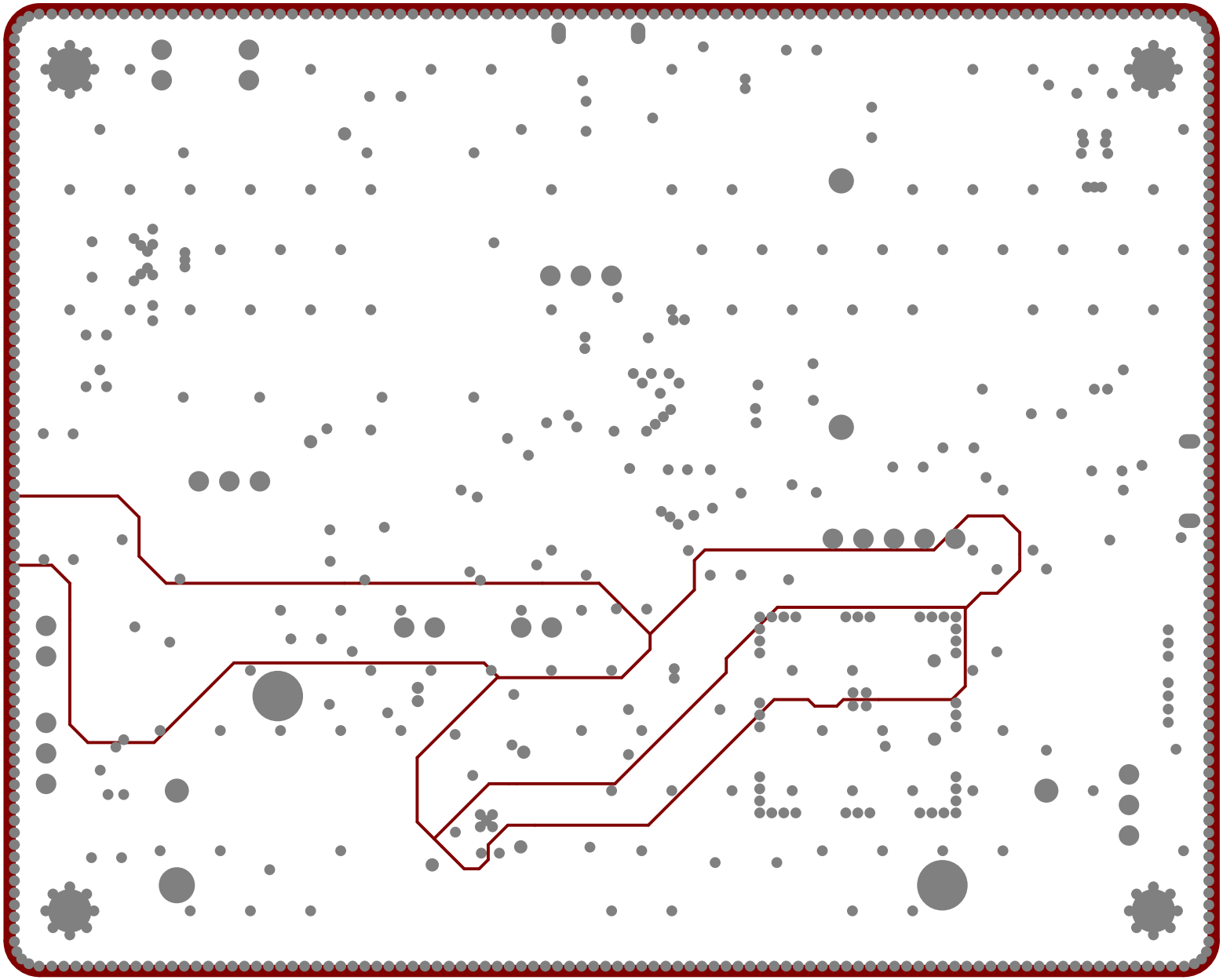


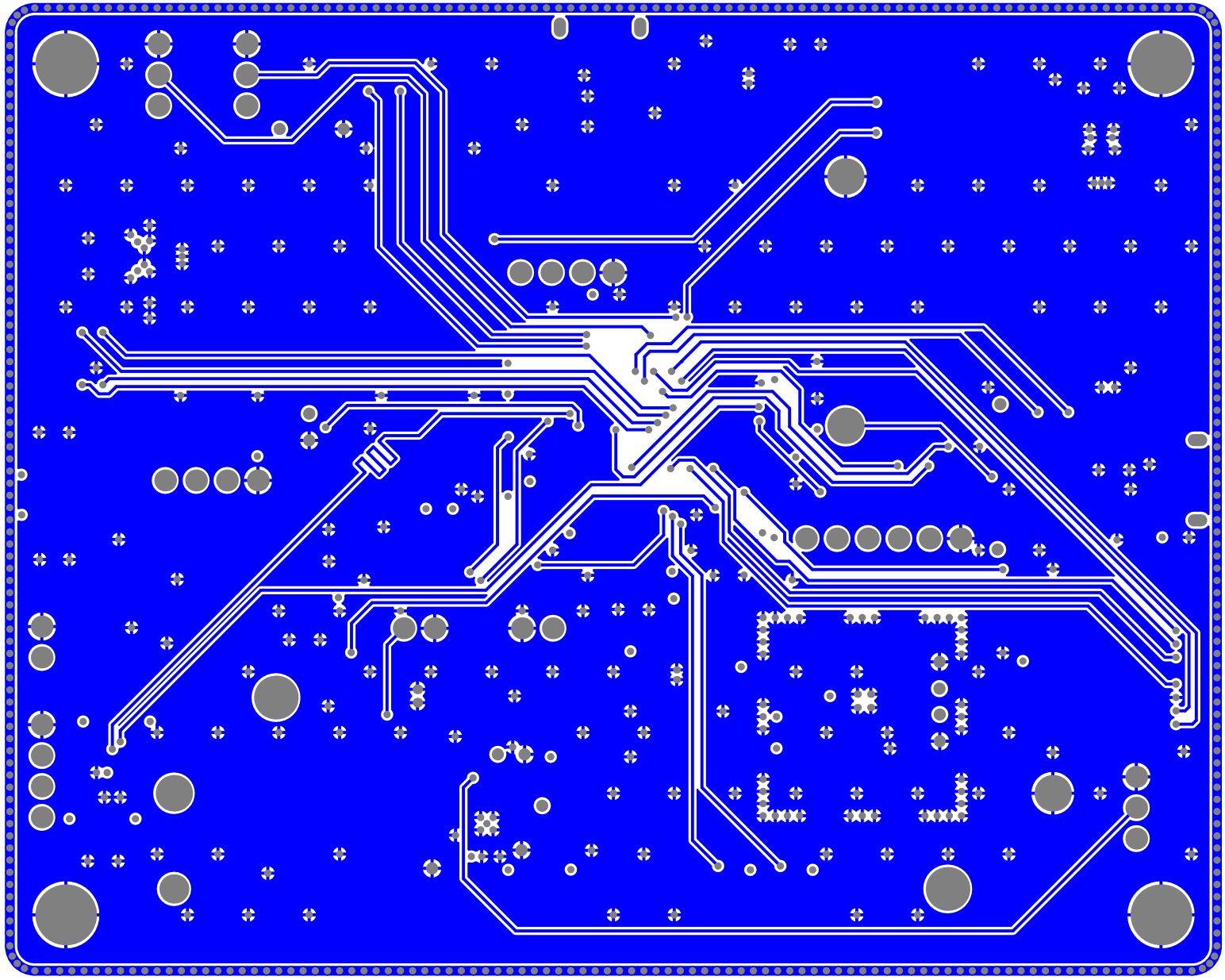
ČVUT FEL	
Project name	boom_spotter.PrjPCB
Sheet name	unused_module.SchDoc
Date	06.04.2021

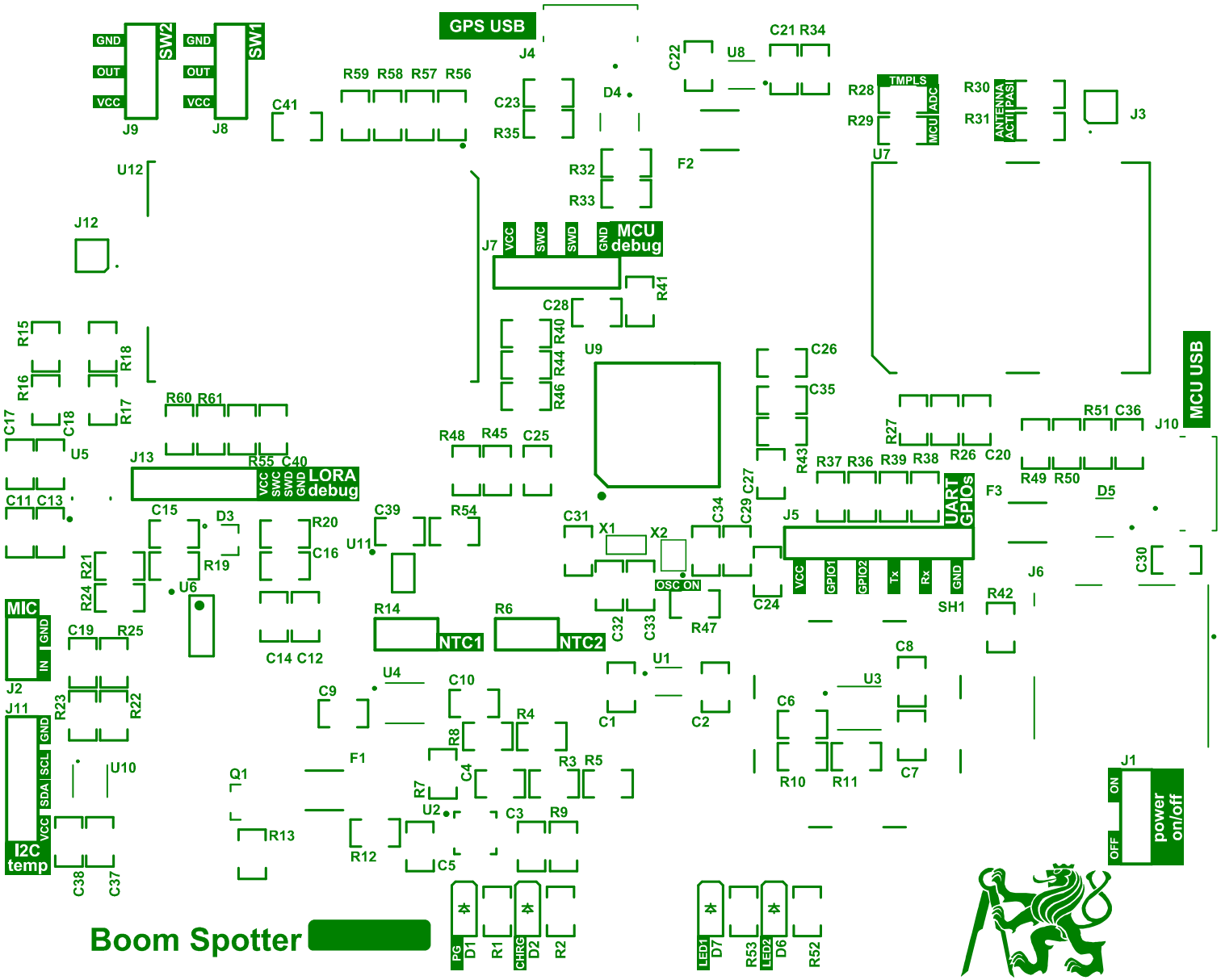

GPS USB
SW2
SW1
GND
OUT
VCC
MCU debug
SWC
SWD
LORA debug
MCU USB
UART GPIOs
GPIO1
GPIO2
Tx
Rx
OSC ON
NTC1
NTC2
MIC
IN
I2C temp
SCL
SDA
power on/off
ON
OFF
TMPLS
MCU ADC
ANTENNA ACTI PASI
PG
CHRG
LED1
LED2
Boom Spotter










GPS USB
SW2
SW1
J9
J8
J4
D4
C41
R59 R58 R57 R56
C23
R35
C22
U8
C21 R34
R28
R29
TMPLS
MCU ADC
R30
R31
ANTENNA ACTI PASI
J3
U7
F2
R32
R33
U12
J12
J7
VCC SWC SWD GND
MCU debug
R41
C28
R40 R44 R46
U9
C26
C35
R43
R27
R51 C36
J10
MCU USB
R15
R16
R18
R17
C17
C18
U5
R60 R61
R48 R45 C25
J13
R55 C40
VCC SWC SWD GND
LORA debug
C11 C13
C15
D3
R20
C16
C39
R54
U11
C31
X1
X2
C34
C29
C27
J5
UART GPIOs
VCC GPIO1 GPIO2 Tx Rx GND
R37 R36 R39 R38
R26
C20
R49 R50
D5
F3
C30
J6
R21
R24
R19
U6
OSC ON
C24
SH1
R42
MIC
GND
IN
J2
C19
R25
C14 C12
R14
NTC1
R6
NTC2
C32
C33
R47
U1
C8
U4
C10
U3
C9
C1
C2
C6
R4
R8
R23
R22
R3 R5
R10
R11
C7
J11
GND
SCL
SDA
VCC
I2C temp
U10
F1
Q1
R7
C4
U2
C3
R9
R13
R12
C5
C38
C37
J1
ON
OFF
power on/off
PG
D1
R1
CHRG
D2
R2
LED1
D7
R53
LED2
D6
R52
Boom Spotter

GPS USB
SW2
GND
OUT
VCC
SW1
GND
OUT
VCC
P2
MCU
debug
GND
SWD
SWC
VCC
MCU USB
LORA
debug
GND
SWD
SWC
VCC
UART
GPIOs
GND
RX
TX
GPIO2
GPIO1
VCC
NTC2
NTC1
P1
power
on/off
ON
OFF
MIC
GND
OUT
I2C
temp
GND
SCL
SDA
VCC

Appendix B

Development Environment and tools

Software	Version
Atollic TrueSTUDIO	9.3.0
arm-atollic-eabi-gcc	6.3.1
arm-atollic-eabi-objcopy	2.27.90.20170215
STM32CubeMX	6.1.1
ARM.CMSIS	5.6.0
FATFS	R.12c
Firmware Package STM32Cube FW_F4	1.25.2
Driver for bme280	3.4.3

Table B.1: Board firmware development environment and tools.

Software	Version
MariaDB	10.5.8
Go	1.16.2
VSCodium	1.55.2
DBeaver	7.3.3.20201231610

Table B.2: Server application development environment and tools.

Software	Version
Matlab	R2020b

Table B.3: Development environment and tools for algorithms.