AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF COMPUTER SCIENCE, ELECTRONICS AND TELECOMMUNICATIONS

DEPARTMENT OF TELECOMMUNICATIONS

Engineering Thesis

Ultra-low power wireless sensor with LoRa radio transceiver

Author: *Michał Markiewicz*
Degree programme: *Electronics and telecommunications*
Supervisor: *dr hab. inż. Kamil Staszek*

Kraków, 2023

Warned of the criminal liability as set forth in Article 115 section 1 and 2 of the Act of 4th February 1994 on copyright and related rights (uniform text Polish Journal of Law [Dziennik Ustaw] from 2019 item 1231 as amended): "Who appropriates the authorship or misleads as to the authorship of the entirety or part of somebody else's work or artistic work shall be subject to a fine, restriction of freedom penalty or of imprisonment up to 3 years. The same penalty shall be imposed on one who distributes (without acknowledging the author by their name or pseudonym) someone else's work in the original version or in an adapted form, the artistic work or who publicly alters such work, artistic work, sound record, videogram or transmission", and also warned of the disciplinary responsibility as set forth in Article 307 section 1 of the Act of 20th July 2018 – Law on higher education and science (uniform text Polish Journal of Law [Dziennik Ustaw] from 2021 item 478 as amended), hereinafter referred to as the Act, "A student shall be subject to disciplinary responsibility for the infringement of university regulations or for acts detrimental to a student's dignity." I created this final diploma thesis by myself and independently and I have not used sources other than the ones acknowledged in the final diploma thesis; The final diploma thesis is a result of my work and does not infringe the copyright of other people; This final diploma thesis does not contain information subject to protection as set forth in the regulation on the protection of classified information.

I would like to thank Pascal Urard and prof. Thomas Skotnicki for inspiring me to explore the topic of ultra-low power wireless communication devices and systems

Contents

1. Introduction

This work describes an ultra-low power wireless sensor developed and programmed by the author as well as backend platform for data storage and visualisation. The original work presented in this thesis includes:

- Implementation of firmware enabling ultra-low power mode (down to several μA).
- Design and implementation of an original energy harvesting circuit.
- Design of a printed circuit boards to evaluate firmware and hardware performance.
- Measurements of power consumption in sleep and active states.
- Design and implementation of backend for data storage and visualisation.

The designed device was intended to perform indoor temperature measurements and send the data via radio in regular time intervals. The requirements regarding the designed sensor are as the following:

1. To measure indoor temperature in range from $-10\,^{\circ}\mathrm{C}$ to $40\,^{\circ}\mathrm{C}$.
2. The measurement accuracy should be at least $0.5\,^{\circ}\mathrm{C}$.
3. To wirelessly transmit the measurements in regular intervals between 30 and 300 seconds.
4. To be a battery powered device, or use an energy harvester.
5. The device must transmit state of charge of the battery.
6. Each device must have a unique identifier.
7. The device should encrypt the data before transmission.

The thesis is organised as following: in chapter 2 the technologies used in this work are described. In chapter 3 implementation details are provided. Chapter 4 contains description of an

iterative prototyping process which lead to creation of several devices with different capabilities. In chapter 5 performance evaluation of the developed devices is given. Chapter 6 concludes the thesis. The appendix contains source codes of firmware and backend as well as server side scripts.

2. Technologies

This chapter contains description of key techologies used in this thesis, namely ultra-low power microcontrollers, long range, low power communication, digital temperature sensor, and serial communication protocols that enable communication between sensor modules.

2.1. Ultra-low power microcontrollers

Modern microcontrollers support switching between low and high power consumption states [1, 2, 3]. This can be used in sensor application when it is not needed to keep the microcontroller active all the time, but only for a short period when the communication is established or when analog-to-digital conversion is performed [4]. Devices working within a wireless sensor network (WSN) could be powered from mains, battery or energy harvester. To increase device autonomy an energy harvester dedicated to WSN can be used [5]. One of the most challenging environments is indoor energy harvesting mainly due to limited access to energy sources [6]. For that reason there is a lot of interest in firmware and hardware development improving energy harvested wireless sensor network devices (EH-WSN) [7].

2.2. Wireless communication

To enable connectivity the following radio standards were considered:

– IEEE 802.15.1 – Bluetooth.

– IEEE 802.11 – WiFi.

– IEEE 802.15.4.

– LoRa.

The comparison of these radio communication technologies are given in Table 2.1.

Table 2.1. Comparison of radio communication technologies [8, 9].

Attribute	Bluetooth	WiFi	802.15.4	LoRa
Frequency [GHz]	2.4	2.4, 5	2.4	0.1-1
Modulation	GFSK	BPSK, O-QPSK	BPSK, QPSK, QAM	CSS
Spreading	FGSS	DSSS, CCK, OFDM	DSSS	CSS
Nominal range [m]	10	100	10	2000
Nominal TX power	0-10 dBm	-41.3 dBm/MHz	-25 – 0 dBm	20 dBm
Throughput	2 MBps	100 MBps	250 kbps	20 kbps
IEEE std.	802.15.1	802.11	802.15.4	None
Power consumption	Low	High	Low	Low

WiFi communication was not well suited for this application due to high power consumption. The initial tests were performed with IEEE 802.15.4 radio modules. Unfortunately, the range was unsatisfactory – the receiver was not able to pick up the signal from the transmitter when both devices were placed in adjacent flats of the same building. Similarly with the Bluetooth. For that reason LoRa radio communication was selected to be used in the design.

2.3. Long range communication

According to the datasheet and application note [9, 10] modulation scheme used in LoRa is a derivative of Chirp Spread Spectrum modulation (CSS). This kind of modulation was orignally developed for radar applications in the 1940's. It was and it is still used in military communications applications. This modulation technique is characterised by low transmission power requirements and robustness from channel degradation mechanisms such as multipath, Doppler or fading. For that reasons it found numerous applications in data communications. LoRa provides a physical layer implementation of CSS and requires higher level layers to be additionally implemented by developers. Modulation and demodulation concepts within LoRa are presented in Fig. 2.1 and 2.2.

In LoRa modulation spectrum spreading is achieved by generation of a chirp signal that varies in frequency. LoRa link budget depends on spreading factor (SF) which can take value in range SF = 7-12. For modulation badwidth BW given in MHz, the modulation bit rate R_b is defined as

$$R_b = SF \frac{1}{\left(\frac{2^{SF}}{BW}\right)} \text{[bits/sec]}$$

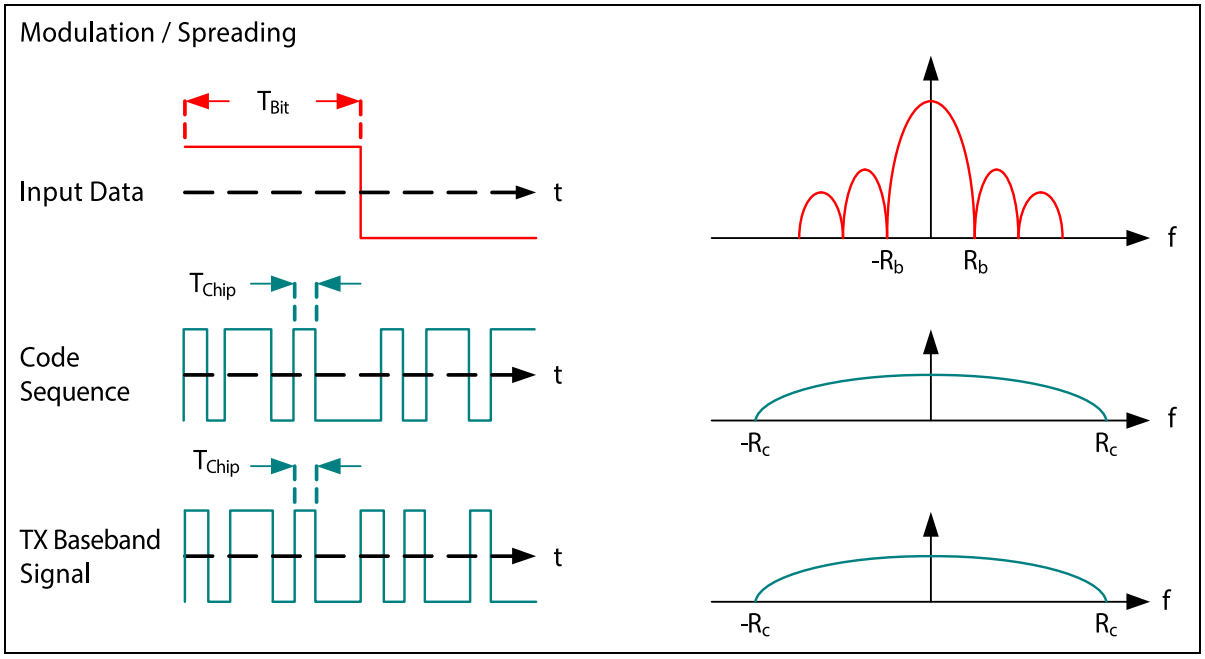

Figure 2.1. LoRa modulation processes [10]

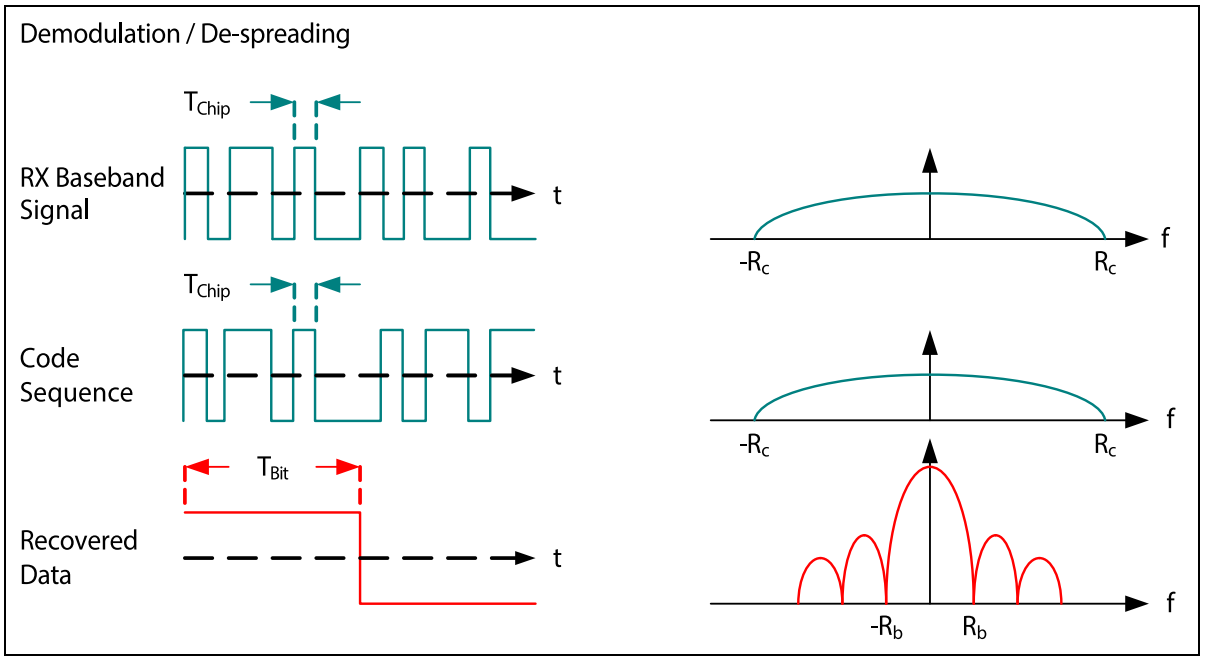

Figure 2.2. LoRa demodulation processes [10]

Link budget – defined as a measure of all gains and losses from the transmitter via the transmission line to the receiver – depends on configurable spreading factor SF parameter. For SF=6, LoRa has -118 dBm, for SF=10:-132 dBm, and for SF = 12, LoRa – excellent -137 dBm.

2.4. Temperature measurement

Temperature sensors are designed to represent information about temperature of a measured object in a form of electrical signal. Some approaches to temperature measurements can be classified as following [11]:

- Resistive
- Infrared
- Thermocouples
- Bi-metallic
- Semiconductor (p-n junction)
- Thermographic liquid crystal
- Pyrometry

Conversion of electrical signal to digital form requires some form of analog to digital converter (ADC). Calibration and adjustment of both temperature measuring element and ADC can be very challenging, especially with additional limitations regarding PCB size, power consumption and bill of material cost. For that reason mass produced integrated circuits with factory calibrated temperature measuring elements and serial communication interface are very popular. Architecture of a sample digital temperature sensor is shown in Fig. 2.3. We see that from embedded designer perspective using this type of modern circuit simplifies the design – all what is needed to get the information about the temperature is to power up the device and provide a serial communication interface.

2.5. Serial communication

To communicate between elements of the embedded system while minimizing number of wires between electronic devices a serial communication protocol can be used. In contrast to parallel communication – where bits are sent over serveral parallel channels – in serial communication data is sent one bit at a time over a single channel. There are three interfaces widely adopted by the market: SPI (Serial Peripheral Interface), I^2C (Inter-Integrated Circuit), and UART (Universal Asynchronous Receiver-Transmitter).

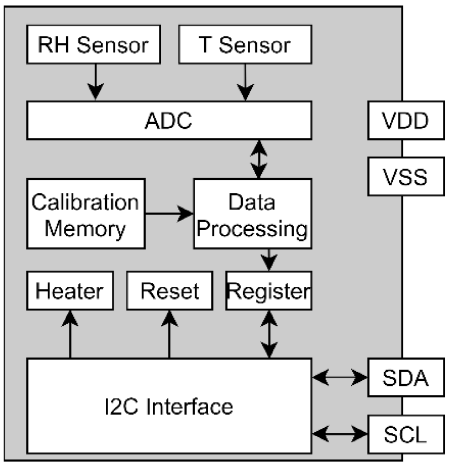

Figure 2.3. Functional block diagram of a digital temperature sensor [12]

2.5.1. SPI Interface

The main features of the SPI interface are:

– SPI connection requires four signals: clock (SCLK), chip select (CS), master output–slave input (MOSI), master input–slave output (MISO).

– Communication is always initialized by the master by pulling CS low.

– Clock (SCLK) is continously generated by the master when CS is low.

– Only a single master is allowed.

– Multiple slaves can be connected to the master via shared SCLK, MOSI and MISO pins.

– In multiple slave configuration each slave requires separate CS pin.

Fig. 2.4 shows configuration with a single slave device.

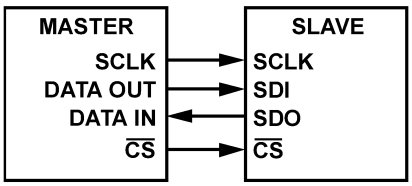

Figure 2.4. Single-Slave SPI configuration [13]

2.5.2. I²C Interface

The main features of the I²C bus are:

– Two bus lines are required: serial clock line (SCL) and serial data line and (SDA).

– Bus lines are bidirectional (open drain connections with pull-up resistors).

– Each device on the bus must have a unique 7-bit address.

– Bus allows multiple slaves and as well as multiple masters.

Fig. 2.5 shows bus topology with a single master and multiple slaves.

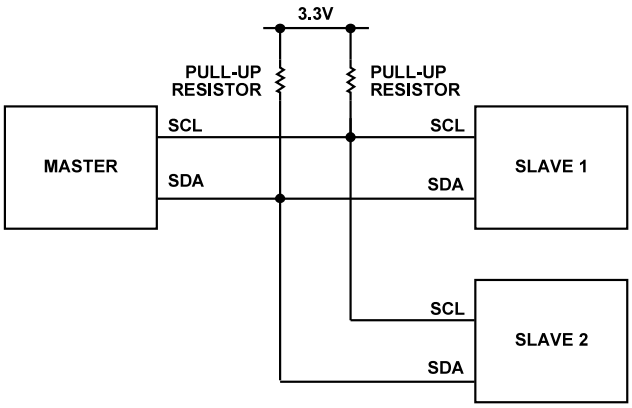

Figure 2.5. Single Master Multi-Slave I²C Bus topology [14]

All transfers are initiated by the master which sends start condition (SDA line goes low during a high period of the SCL line) and ends with stop condition (SDA goes high during a high period of the SCL line). After the start condition, the master sends a byte on the SDA line, along with eight clock pulses on SCL line. The first seven bits of the first byte is the 7-bit slave address (unique on the bus). The slave responds with acknowledge (ACK) if this 7-bit address matches the address of the slave device. The eighth bit (LSB) denotes the direction of the message. If it is set, then the slave should send the data to the master [14]. A typical I²C communication sequence is shown in Fig. 2.6.

2.5.3. UART Interface

The main features of the UART interface are:

– Asynchronous data transmission requires no clock signal to synchronize the output of bits from the transmitting device to the receiving device.

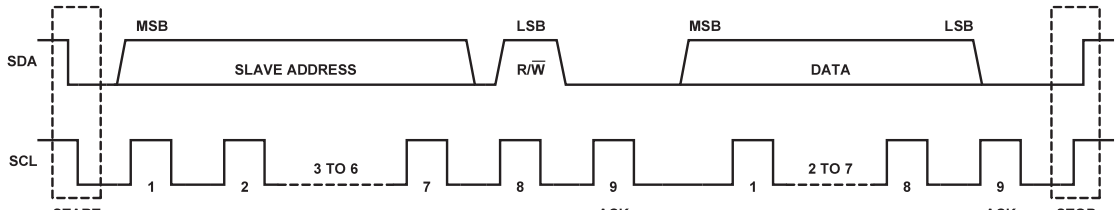

Figure 2.6. Typical I^2C communication sequence [14]

– Requires only two wires (plus ground).

– Does not support multiple slave or multiple master: only two devices can be connected to each other.

Fig. 2.7 shows typical UART setup.

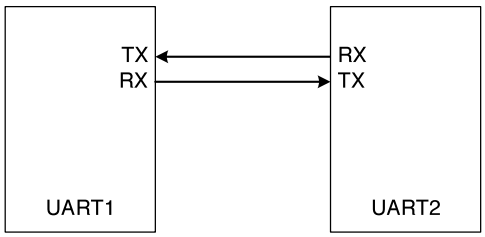

Figure 2.7. Typical UART setup [15]

A typical UART communication sequence is shown in Fig. 2.8. It is initiated by a falling edge of the start bit followed by data bits, parity bit and stop bits. This process repeats until all data bits are transmitted, and then the rising edge of the stop bit returns the line to its idle state. The number of data, parity and stop bits is configurable. At most eight data bits can be send at once. Presence of a parity bit helps in error detection.

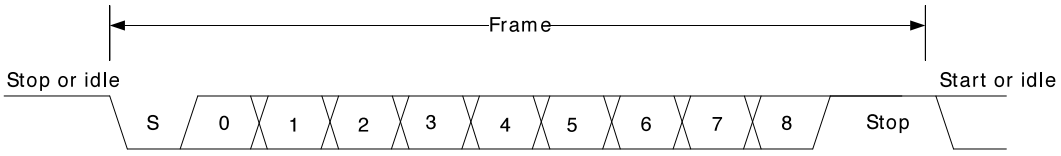

Figure 2.8. Typical UART communication sequence [15]

3. Implementation

3.1. System architecture

The main idea motivating this thesis is to measure indoor temperature, transmit the data and present it to the end user. Fig. 3.1 shows a simplified diagram of information flow within the system. Temperature measurements are send by sensors via radio to a router. The router is made of a Raspberry Pi with a radio receiver extension board connected to it via header. The router has Ethernet interface which is used to transfer the data over the Internet to server with Message Queuing Telemetry Transport (MQTT) service. The data is then routed to a web service which stores them in the database from which they can be further presented to the users via web interface.

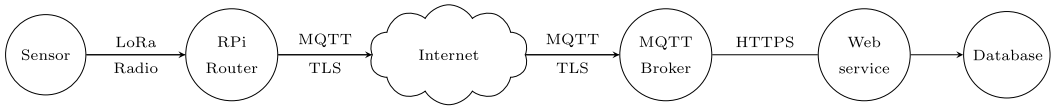

Figure 3.1. System architecture: temperature measurements are wirelessly transmitted to Raspberry Pi, and then sent to web service, which saves them to the database.

3.2. Sensor architecture

General hardware architecture of an wireless sensor node can be split into the following subsystems [4]: *sensing subsystem* (physical measurements and analog to digital conversion), *computing subsystem* (data processing), *communication subsystem* (radio communication), *power subsystem* (providing system supply voltage). In our case temperature is measured by a temperature sensor which internally digitalises readouts and transmits them via I^2C interface to the microcontroller. The microcontroller is responsible for data processing. It assembles a packet with information from the temperature sensor, encrypts it and sends the transmission request

to the radio module via SPI interface. After transmission power supply is cut off from both the temperature sensor and the radio module to save power, and the microcontroller enters a low power mode. Just before that a real time clock is set up and the clock source is changed to a low speed external oscillator. Fig. 3.2 shows information flow within the sensor.

Figure 3.2. Sensor architecture: temperature sensors sends the data to the microcontroller via I^2C interface, the data is then send via SPI interface to the radio module.

3.3. Firmware main loop

The control program can be presented in a form of infinite loop with steps listed in Algorithm 1.

while *true* **do**
    Set up the MCU GPIOs, high frequency clock source, timers, interrupts ;
    Communicate with the temperature sensor via I^2C interface;
    Encrypt the data and prepare a packet to be sent;
    Communicate with the radio module via SPI interface;
    Switch to a low frequency clock source. Setup a real time clock (RTC);
    Enter a low power mode;
    Wait for RTC interrupt;
end

Algorithm 1: Sensor main loop

After connecting power, and initial set up, the device enters an infinite loop in which it configures general purpose intput outpus (GPIOs), timers, clock source and interrupts. Then it reads information about temperature from the temperature sensor via I^2C interface. Then the microcontroller processes the information by assembling a radio packet with encrypted information about the temperature. Then the packet is send via SPI interface to the radio module, which wirelessly transmitts the packet to the receiver. After that a real time clock (RTC) is set to wake up the sensor in a couple of minutes. The device enters a low power mode, where all the peripherals like temperature sensor and radio module are turned off and the microcontroller is clocked from a low frequency external oscillator to conserve power. When RTC generates an interrupt it causes the loop to be repeated.

Table 3.1. Comparison of different microcontrollers used in designed devices

Attribute / Microcontroller	STM32L072CB	STM32L151C8	STM8L051F3
Sleep with RAM and RTC [μA]	0.86	1.2	1.3
Power supply [V]	1.65–3.6	1.65–3.6	1.8–3.6
RAM [kB]	20	32	1
Flash [kB]	64	32	8
Price [USD @ 10kU]	2.39	2.34	0.49

Source codes implementing temperature measurements and radio transmission are given in the appendix.

3.4. Low power design

The microcontroller executing the main loop spends most of the time in a low power mode waiting for a RTC interrupt. To conserve the power it is necessary to select a microcontroller with as low sleep power consumption as possible. The other factors that have to be taken into account in MCU selection are: size of RAM and Flash, voltage range, and unit price. Table 3.1 shows a summary of attributes of selected STMicroelectronics microcontrollers.

For further investitagion all these MCUs were used in prototypes. Correctly implemented behaviour of the main loop is shown in Fig. 3.3. Power consumption spikes are associated with the sensor activity. During sleep mode the device is drawing a couple of micro-amperes. To conserve power even more, an additional circuit might be added in a form of transistor key that cuts off the power completely to the periperals such as temperature sensor and radio module.

3.5. Energy harvesting

To increase sensor autonomy, i.e. to extend intervals between battery replacement it is possible to add an energy harvesting unit. It might be in a form of a dedicated integrated circuit, like STMicroelectronics SPV1050, or – a custom made circuit build from several discrete components like diodes, comparators, and inverters.

The architecture of the designed power management unit is shown in Fig. 3.4. An amorphous photovoltaic cell optimised for indoor lighting is connected to a charge pump circuit. A square wave required to increase input voltage is generated by three inverters connected in series. Inverters are powered by the photovoltaic cell, so they do not draw energy from the battery in

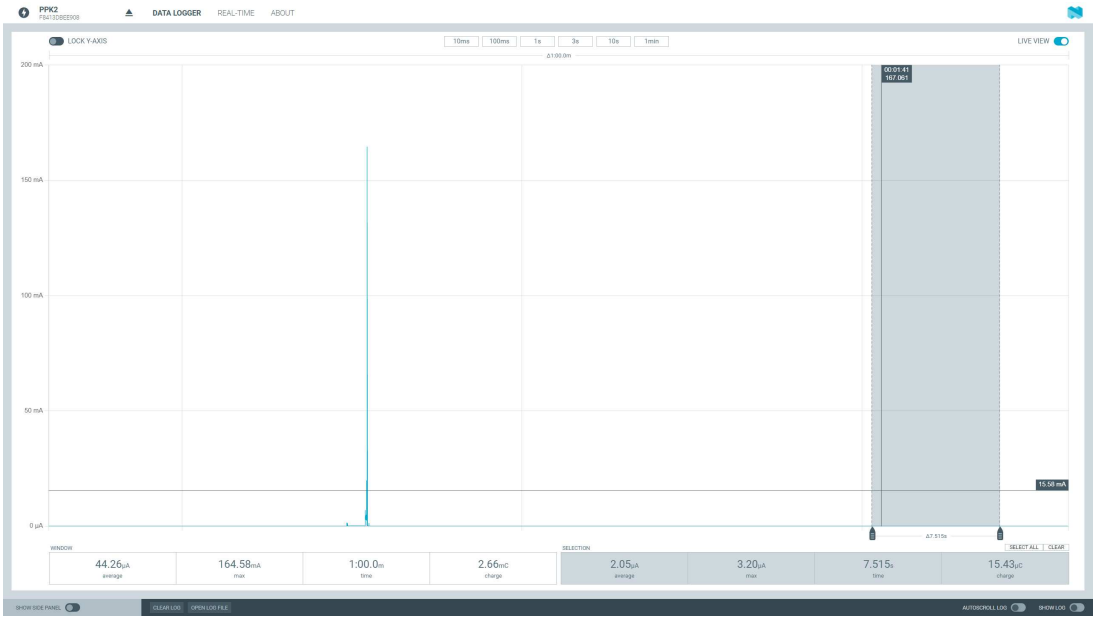

Figure 3.3. Power profile during activity (spike) and sleep mode

low-light conditions. To protect the battery from overcharging a low-power comparator is used. It has only 600nA of quiescent current [16]. It is powered from a low dropout regulator (LDO) connected to the battery. LDO provides also reference voltage (adjusted to proper level by a voltage divider) to the comparator. When the battery voltage exceeds a predefined threshold – then comparator output goes low disconnecting charge pump output from the battery. This solution does not provide the optimal efficiency of energy harvesting (specifically it is lacking of MPPT) but it is good enough to ensure positive power balance of the device.

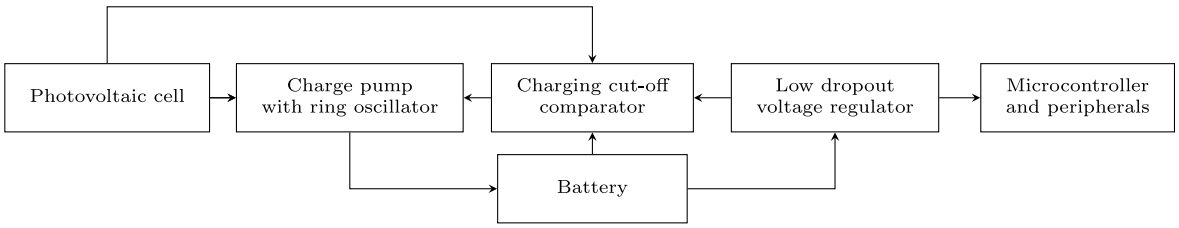

Figure 3.4. Architecture of the custom made power management unit

3.6. Radio receiver

Wireless data transmitted by the senors have to be first received to be then stored on the server side. The wireless router has been constructed from a Raspberry PI, by connecting a custom made extension board to RPi 40-pin header. This header is shown in Fig. 3.5. It provides

power supply for any extension board connected to it, as well as UART communication interface. The firmware for the the extension board was adapted from the sensor firmware. Its main loop is shown in Algorithm 2.

```
while true do
    Communicate with the radio module via SPI interface;
    if packet received then
        Read packet ;
        if packet is correct then
            Decrypt packet;
            Parse packet content ;
            Transmitt packet content via UART interface ;
        end
    end
end
```

Algorithm 2: Receiver main loop

The microcontroller communicates with the radio module over SPI interface. When it receives information about packet awaiting in receiving buffer, it reads packet contents and check its correctness (packet size). If the packet is OK, it is decrypted and parsed. A character string is then constructed with the received values and send to the UART interface. UART pins are connected to RPi processor, so there is a need to have a separate script working on RPi to receive the data and transfer them to the server.

Source codes implementing the radio receiver extension board and server scripts are given in the appendix.

3.7. Network configuration

All the servers providing services for reception, storage and visualisation of sensor data were deployed in a form of Linux and Unix virtual machines running under Microsoft Hyper-V Server 2019 supervision. Hyper-V Server Virtual Switch manager was configured to connect all the virtual machines to the external network. Network Address Translation (NAT) objects were created that translate external network addresses to internal network addresses. Table 3.2 shows the summary of virtual server roles. Fig. 3.6 shows Hyper-V Manager window with configured servers list.

Network configuration scripts are given in the appendix.

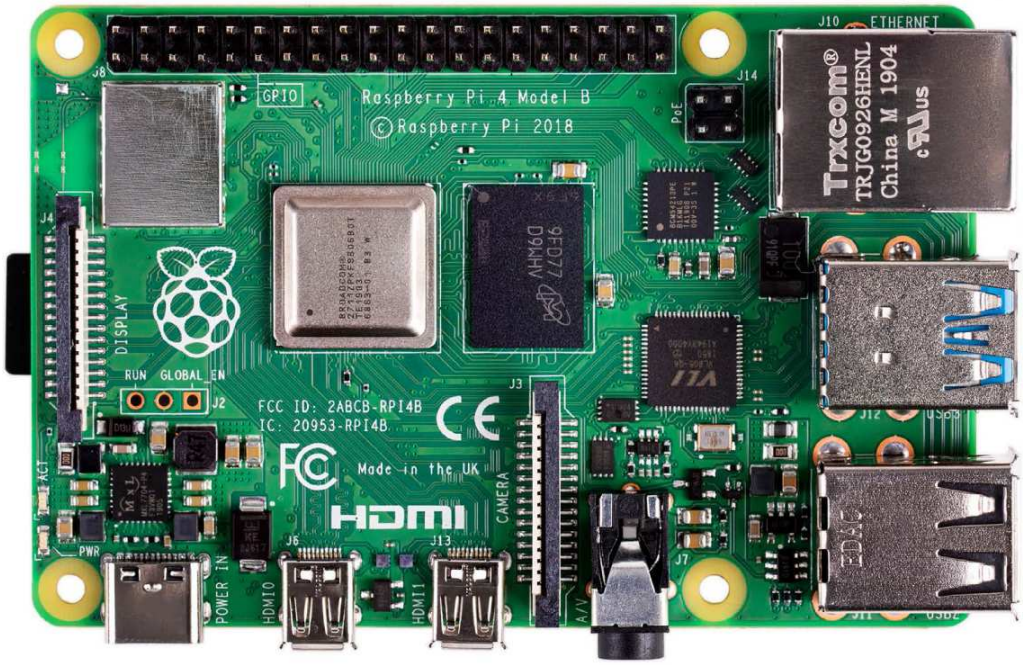

Figure 3.5. Raspberry Pi single board computer with 40-pin header [17]

Table 3.2. Servers used for reception, storage and visualisation of sensor data

Server name	Operating system	Purpose
mq	FreeBSD	MQTT broker
ms	Ubuntu	MSSQL database, Blazor
web	OpenBSD	Application layer gateway

3.8. Web interface

A custom-build web interface provides a user friendly web page that shows list of all sensors connected to the server and provides a detail view of a specific sensor. The whole service follows three layer architecture concept by separating application into a presentation tier, an application tier and a data tier. The presentation tier was implemented in ASP.NET using Razor syntax. The application tier was implemented in C#. The data tier is running on a Relational Database Management System (RDBMS). Clearly defined application programming interfaces (API) allows communication between tiers in easy way. Screens of sensor list and sensor details are shown in Fig. 3.7 and 3.8.

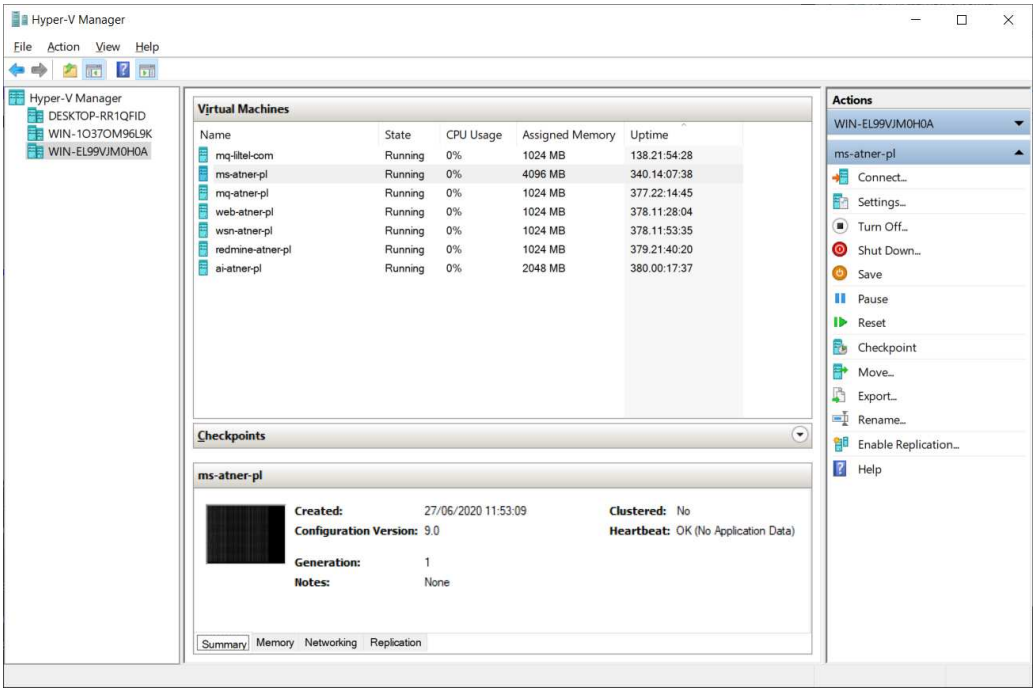

Figure 3.6. Hyper-V manager with configured servers

Source codes implementing the web interface, application layer and database scripts are given in the appendix.

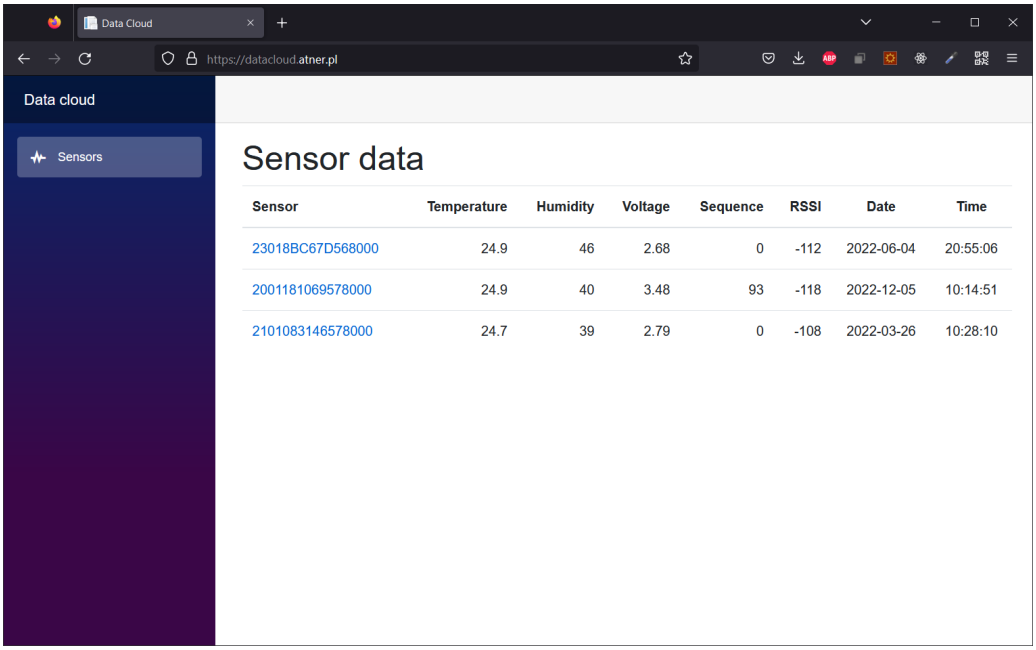

Figure 3.7. Web interface with a list of connected sensors

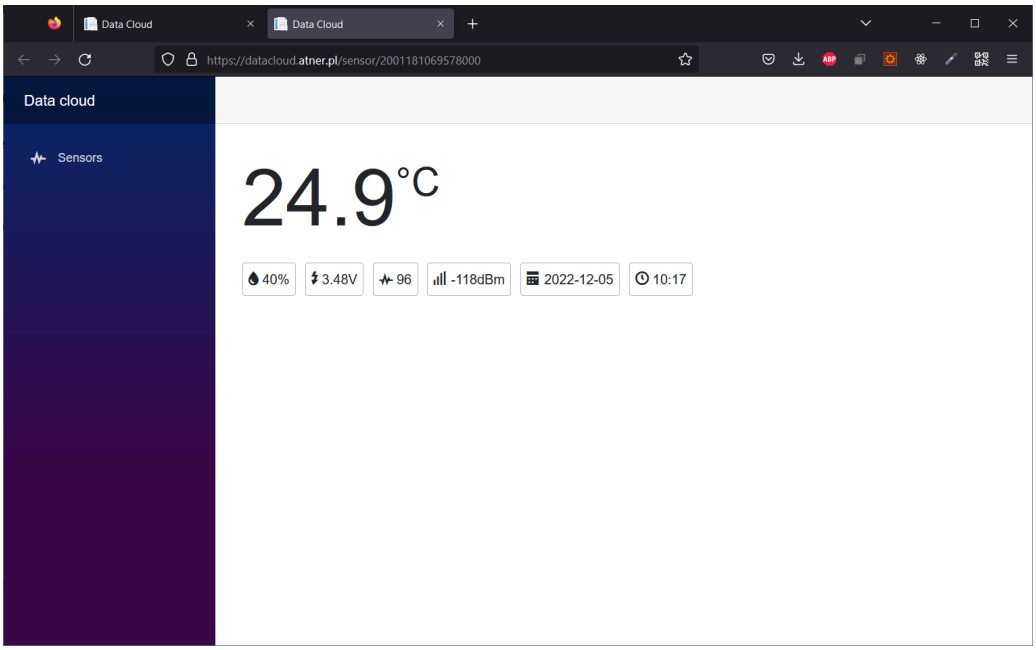

Figure 3.8. Web interface with details of a specific sensor

4. Prototyping

Initially, a battery powered device with LoRa module and SHT30 temperature sensor was designed. STM32L072CBT6 microcontroller produced by STMicroelectronics was selected because of its low power characteristics and USB support (required for device configuration). STM32L072CBT6 is a ultra-low-power System-On-Chip having the following features [1]:

- Ultra-low-power 32-bit MCU Arm-based Cortex-M0+.
- 20 KB RAM, 128 KB Flash, 6 KB of data EEPROM.
- Supply current 0.86 μA in stop mode with RTC and 20 KB RAM retention.
- 1.65 V to 3.6 V power supply.
- $-40\,^{\circ}\mathrm{C}$ to $125\,^{\circ}\mathrm{C}$ operating temperature range.
- Pre-programmed bootloader USB, USART supported.
- USB 2.0 crystal-less, battery charging detection peripheral communication interface.

The radio communication is provided by RA-02 ultra-long distance spread spectrum communication module which has the following specification [18]:

- Frequency range: 410–525 MHz.
- 2.5 V to 3.7 V supply voltage.
- Typical supply current: TX: 93 mA, RX: 12.15 mA.
- $-30\,^{\circ}\mathrm{C}$ to $85\,^{\circ}\mathrm{C}$ operating temperature range.
- Antenna connector: IPEX.
- SPI communication interface.

Measurements are performed by SHT30 digital humidity and temperature sensor produced by Sensirion [19]:

The radio communication is provided by RA-02 ultra-long distance spread spectrum communication module which has the following electrical characteristics [19]:

– 2.15 V to 5.5 V supply voltage.

– Supply current: idle: 0.2-2.0 μA, measuring: 0.6-1.5 mA.

The temperature is measured by SHT30 sensor communicating with the MCU via SPI interface. The whole device is directly powered by IMR18350 – a non-rechargeable lithium-ion battery. Relationship between the most important elements of the first version of the device is shown in Fig. 4.1.

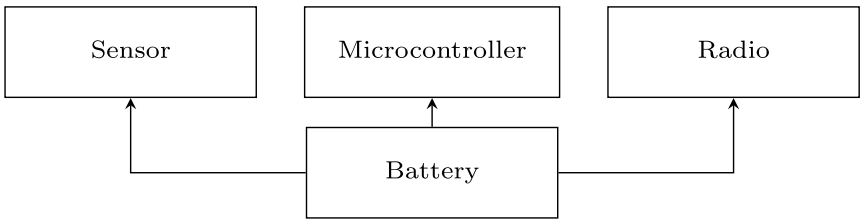

Figure 4.1. Energy flow in the first version of the device

Ten prototypes were manufactured. The designed circuit worked as expected. Due to chip shortage it was hard to obtain desired MCU for future development. For that reason in the next iteration an MCU which was easier to obtain and have less Flash memory was selected. STM32L151C8T6 microcontroller produced by STMicroelectronics was selected because of its low power characteristics, USB support, and availability. STM32L151C8T6 is a ultra-low-power System-On-Chip having the following features [2]:

– Ultra-low-power 32-bit MCU Arm-based Cortex-M0+.

– 10 KB RAM, 32 KB Flash, 4 KB of data EEPROM.

– Supply current 1.2 μA in stop mode with RTC.

– 1.65 V to 3.6 V supply voltage.

– $-40\,^{\circ}\mathrm{C}$ to $85\,^{\circ}\mathrm{C}$ operating temperature range.

– USB peripheral communication interface.

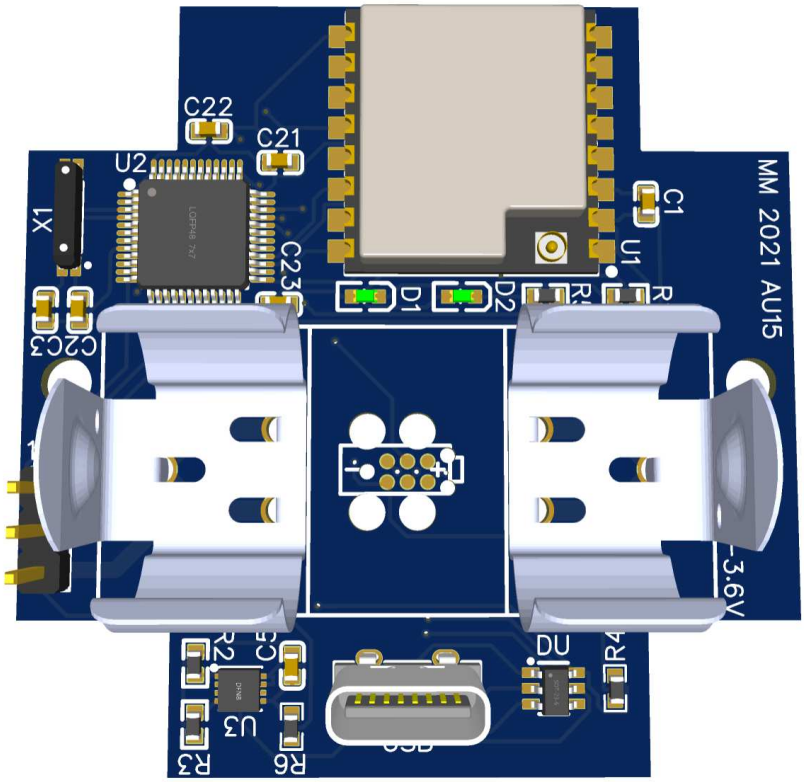

Figure 4.2. Visualisation of the first version of the device

The selected microcontroller requires a high frequency crystal for high speed oscillator (HSE) to handle USB connectivity. For that reason in the second version there are two clock sources: 32.768 kHz quartz for low speed external (LSE) and 16 MHz quartz for HSE. The overall architecture of the device remained unchanged.

Some minor adjustments were introduced regarding programming connector location, and PCB border outline to better fit into enclosure.

Ten prototypes were manufactured. The designed circuit worked as expected. Due to chip shortage it was not possible to obrain RA-02 module. For that reason in the next iteration an equivalent module was added to the PCB: SX1278S47S+T-X1 produced by Vollgo. Moreover, it was necessary to adapt the PCB to a larger enclosure. The major change was the introduction of an energy harvester in a form of dedicated integrated circuit having the following features [20]:

– Ultra-low power energy harvester and battery charger.

– Transformerless PV modules energy harvester.

– Programmable MPPT by external resistors.

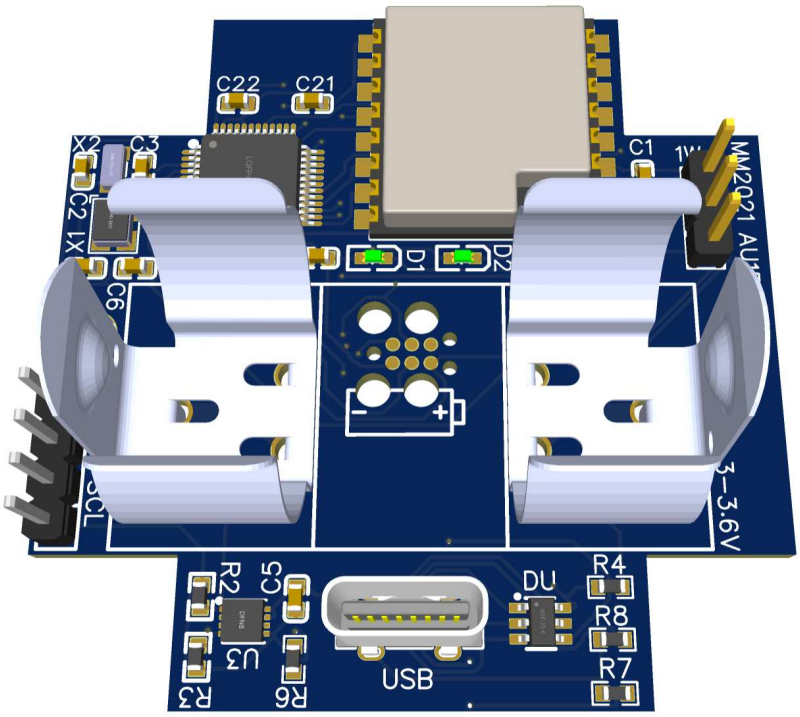

Figure 4.3. Visualisation of the second version of the device

– Two fully independent LDOs (1.8 V and 3.3 V output).

– Fully integrated MOSFETs for boost or buck-boost configurations.

– $-40\,°\mathrm{C}$ to $125\,°\mathrm{C}$ operating temperature range.

The device is intendent to be used indoors. For that reason an amorphous photovoltaic cell – Panasonic AM1820 was selected as the energy harvesting power source. PV cell has the following characteristics with 200lx ($25\,°\mathrm{C}$) fluorescent light source [21]:

– Open circuit voltage V_{OC}=4.9 V.

– Short circuit current I_{OC}=14.5 μA.

– Operating voltage and current 3.0 V–13.3 μA.

Ten prototypes were manufactured. The designed circuit worked as expected regarding radio communication and temperature measurements. However, due to chip shortage it was not possible to purchase SPV1050 to fully populate the PCB. Relationship between the most important elements of the third version of the device is shown in Fig. 4.4.

In the next iteration a purpose build energy harvester was introduced which used more widely available components (a comparator, several diodes, and a low drop voltage regulator) instead of a hardly accessible dedicated energy harvesting integrated circuit. Some minor

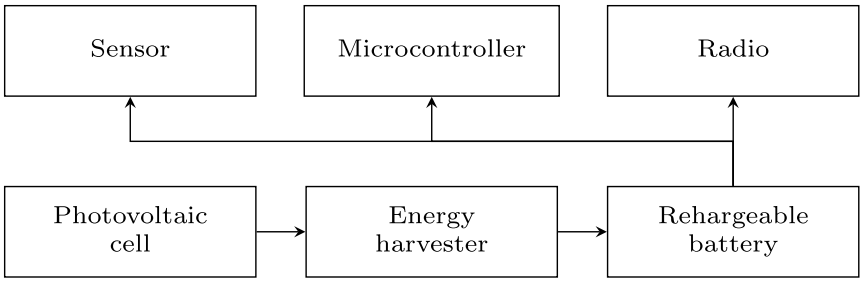

Figure 4.4. Energy flow in the third version of the device

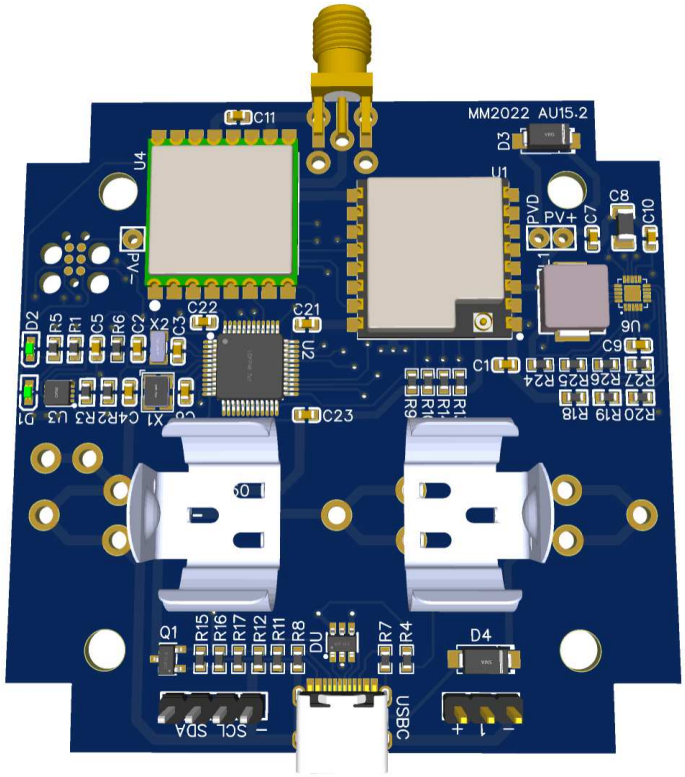

Figure 4.5. Visualisation of the third version of the device

adjustment were introduced to better fit into enclosure. The charging circuit consisted of a comparator powered by a photovoltaic cell, that has non-inverting input connected to the battery (by a voltage divider) and inverting input connected to the low drop voltage regulator (by a voltage divider). Its output is connected by a Schottky diode to the battery. The selected comparator [16] has a sub-microamp quiescent current and push-pull output. When the battery voltage is below predefined cut-off voltage the output of the comparator goes up and directly connect photovoltaic cell to the battery. Photovoltaic cell characteristics is similar to current source, so the battery is then charging. The diode blocks from reverse current flow in low-light conditions.

When the battery voltage exceeds the predefined threshold, the comparator output goes low, so there is no charging, and the diode prevents the battery from discharge.

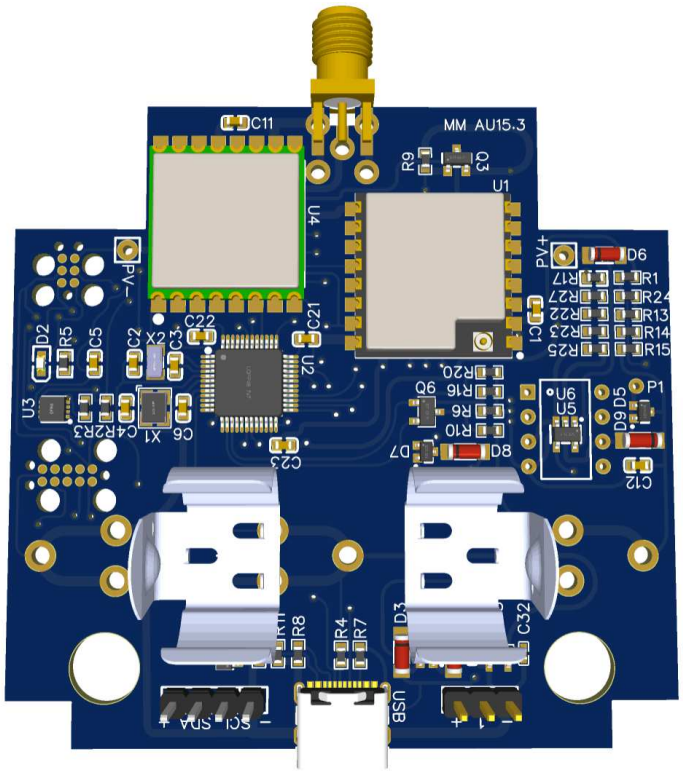

Figure 4.6. Visualisation of the fourth version of the device

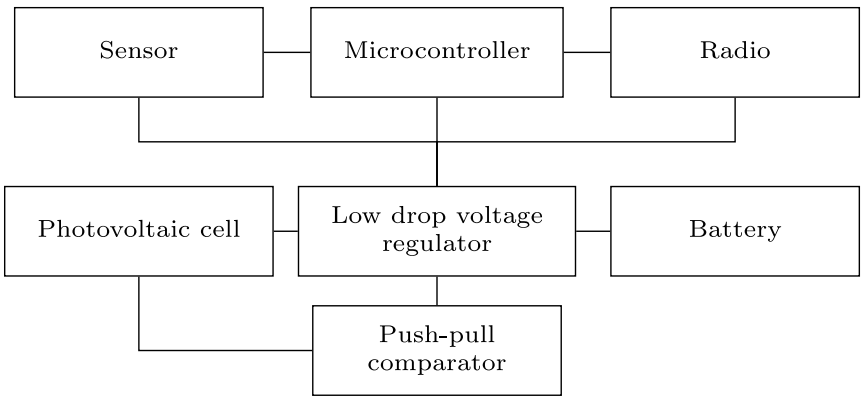

Figure 4.7. Architecture of the fourth version of the device

Ten prototypes have been manufactured. The designed circuit worked as expected regarding radio communication and temperature measurements. Relationship between the most important elements of the third version of the device is shown in Fig. 4.4.

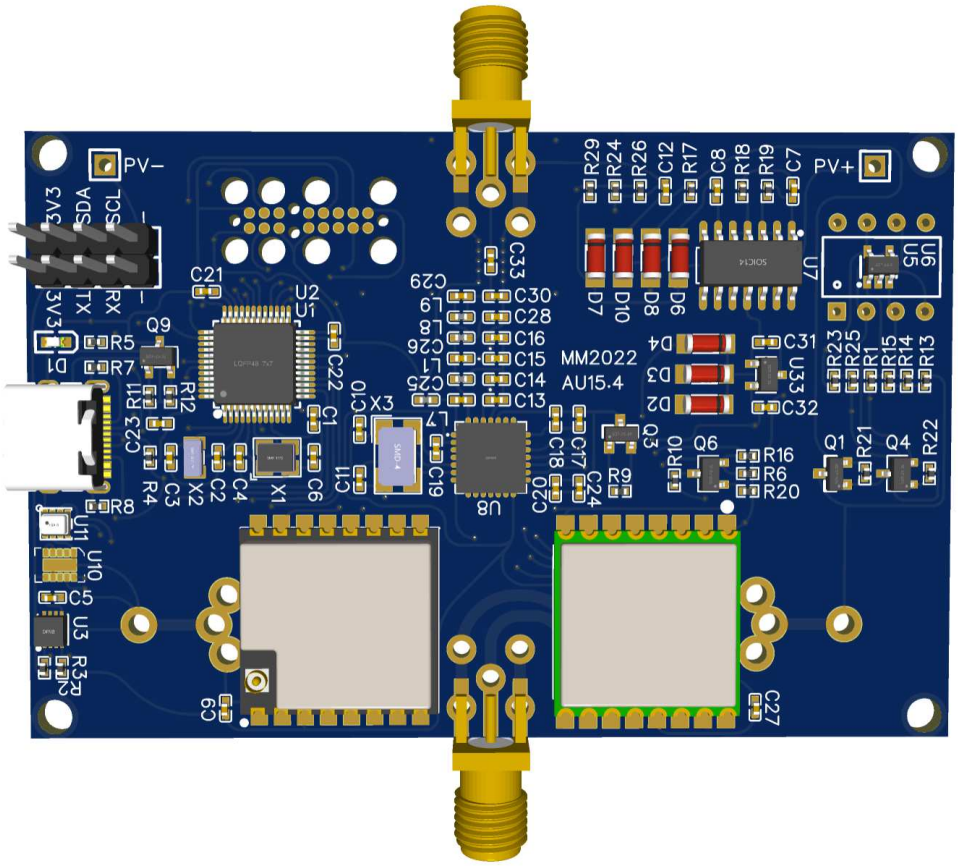

Figure 4.8. Visualisation of the fifth version of the device

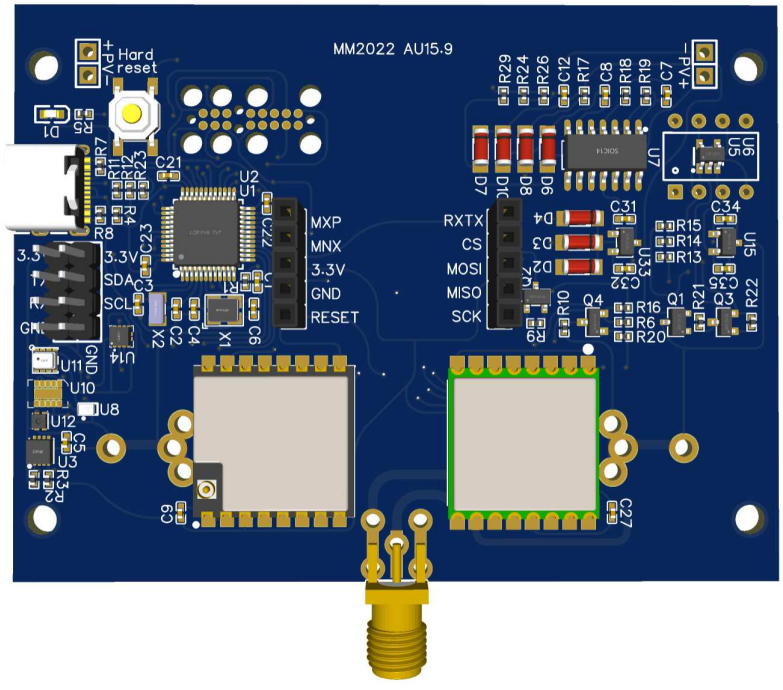

Figure 4.9. Visualisation of the sixth version of the device

In the next iteration an additional connector has been added to support a custom made radio extension board useful for radio communication development.

Finally, the seventh version of the sensor with a different low-power microcontroller from STM8L family was used [22]. The number of parts was reduced to minimize the bill of materials and the total cost. This device was designed to serve a double purpose: to act as a temperature sensor, or after soldering an additional IDC connector to act as an receiver extension board for Raspberry Pi.

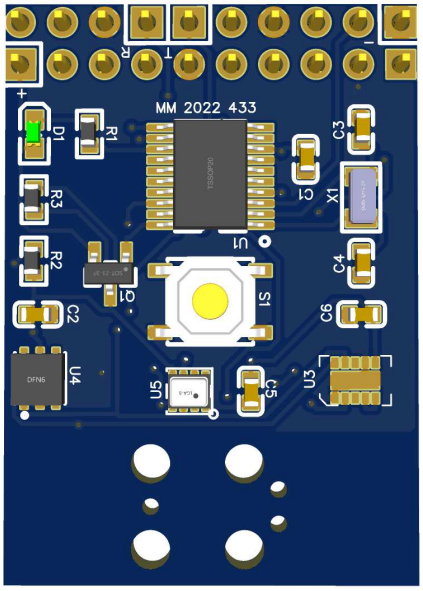

Figure 4.10. Visualisation of the seventh version of the device

5. Evaluation

Devices developed in prototyping phase described in the previous chapter have been assembled and tested. Fig. 5.1 depitcs the fourth version of the device inside Kradex Z-123 enclosure. There were some minor mechanical issues related to enclosure milling for antenna and USB-C connectors. In the fifth version different enclosure was tested – Kradex ZM-125. The PCB fits nicely, there is much more space for the battery as shown in Fig. 5.2. Antenna with IPEX connector allows to provide RF signal outside the enclosure. However, in configuration with SMA connector the PCB turned out to be too narrow. In the sixth version of the device this problem was solved. Moreover, that version also provides an option for a detachable radio for further development, as shown in Fig. 5.3.

Figure 5.1. Prototype with Z-123 enclosure assembled for evaluation

The seventh version of the device is equipped with an IDC connector which allows connecting to the 40-pin header of Raspberry PI and act as a radio receiver. Fig. 5.4 shows the device attached to 40-pin header of IO board with Raspberry PI Compute Module 4.

Figure 5.2. Prototype with ZM-125 enclosure assembled for evaluation

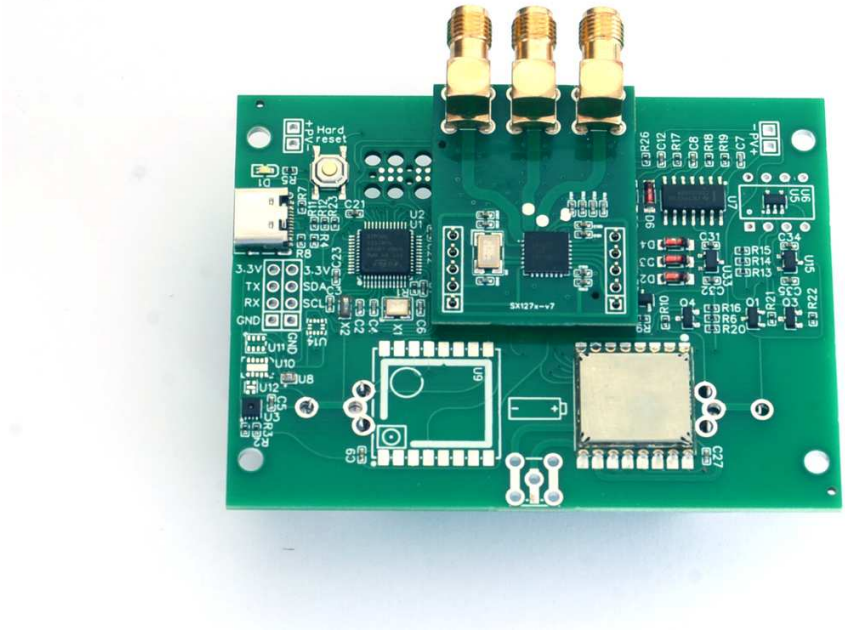

Figure 5.3. Prototype with an additional extension board for further development of wireless communication

5.1. Power consumption measurements

To measure energy consumption levels during sleep and activity a Power Profiler Kit II from Nordic Semiconductors was used. It has the following features [23]:

Figure 5.4. Receiver extension board mounted to 40-pin header of RPi

– Measurement range from 200 nA to 1 A with variable resolution between 100 nA and 1 mA.

– Up to 100k samples per second.

– Voltage source from 0.8 V to 5 V.

The testbed setup is schematically shown in Fig. 5.5. The Device Under Test (DUT) is powered by the Power Profiler Kit II. The results can be displayed on a PC connected via USB interface.

The power profile of the fifth version of the device during sensor activity is shown in Fig. 5.6. The power consumption during sensor activity is similar across all versions of devices. The power profile duirng activity is shown in Fig. 5.7. After wakeup, the device is communicating via I^2C interface with the temperature sensor, then prepares a packet to be send and transmitts it via SPI interface to radio module. Power consuption peak takes place when the radio transmission is active. After transmission the device goes back to the sleep mode with only a real-time clock beign active. The power consumption during sleep was improved during the development achieving 2 μA in the seventh version of the device as shown in Fig. 5.8.

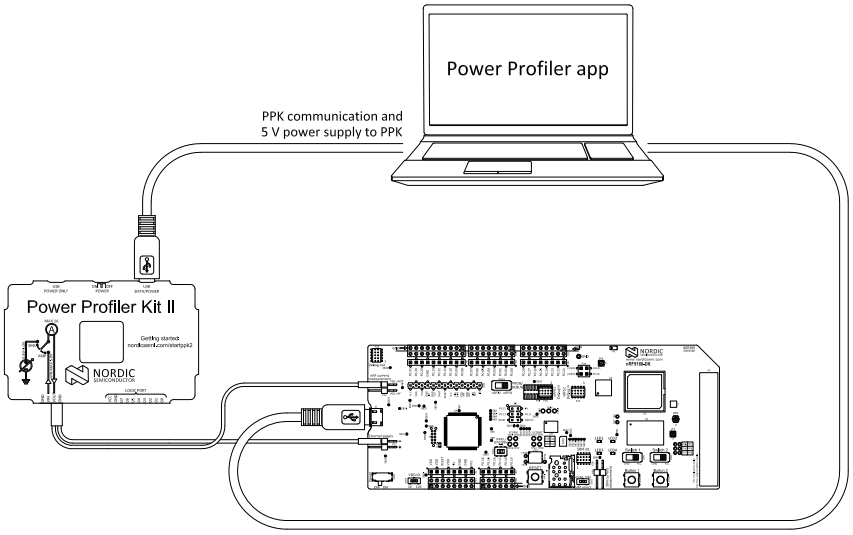

Figure 5.5. Power profiler setup in source meter mode [23]

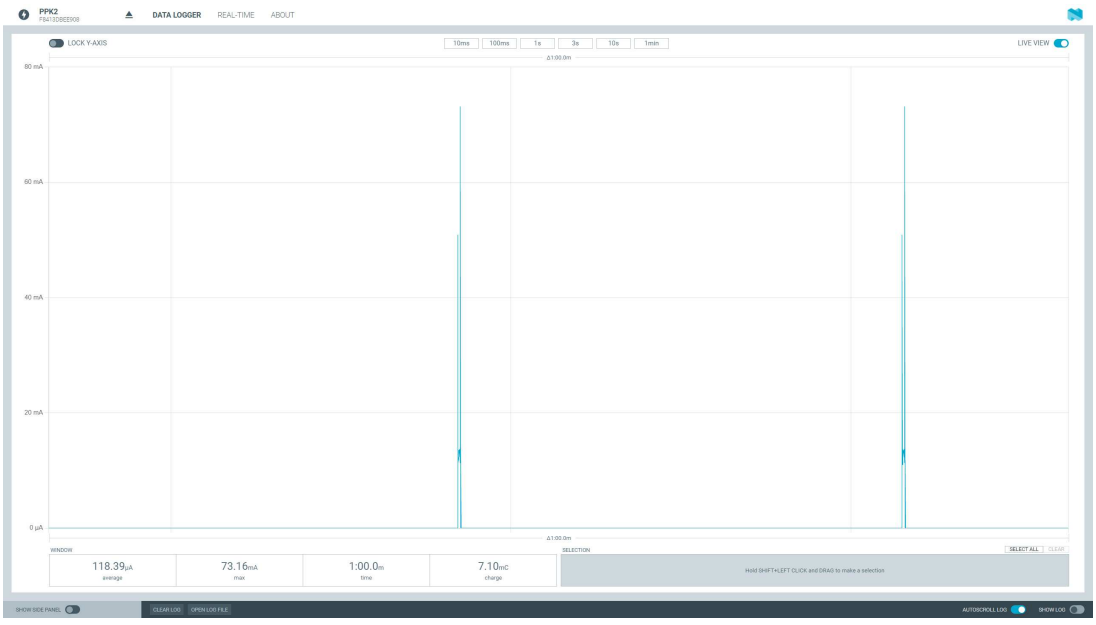

Figure 5.6. Power profile during device deep sleep and activities

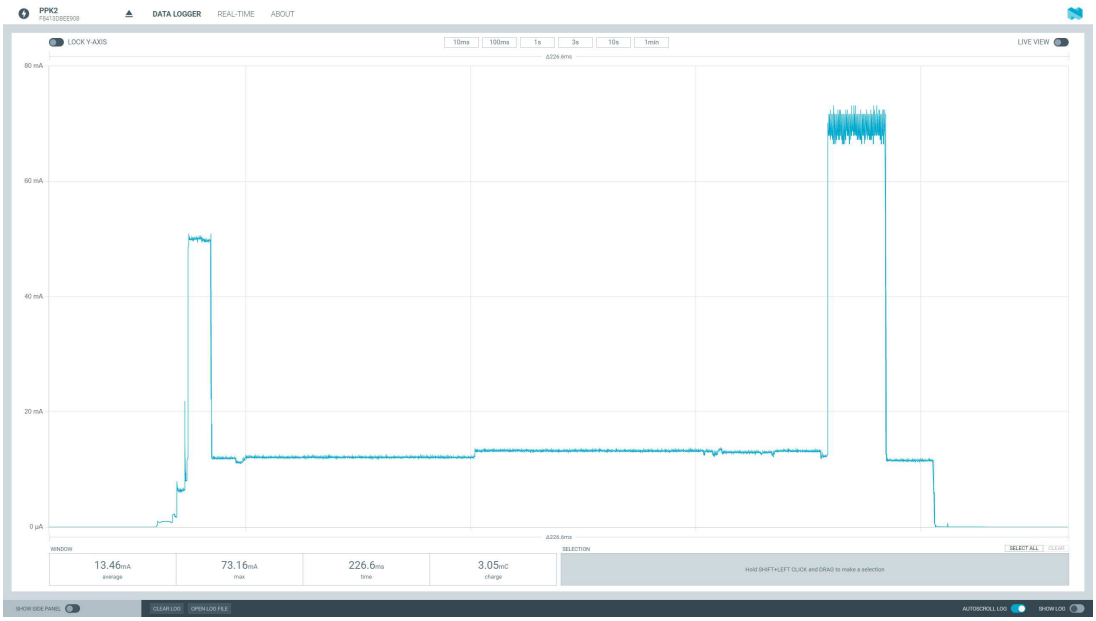

Figure 5.7. Power profile during sensor activity, just after device wake-up from deep sleep mode

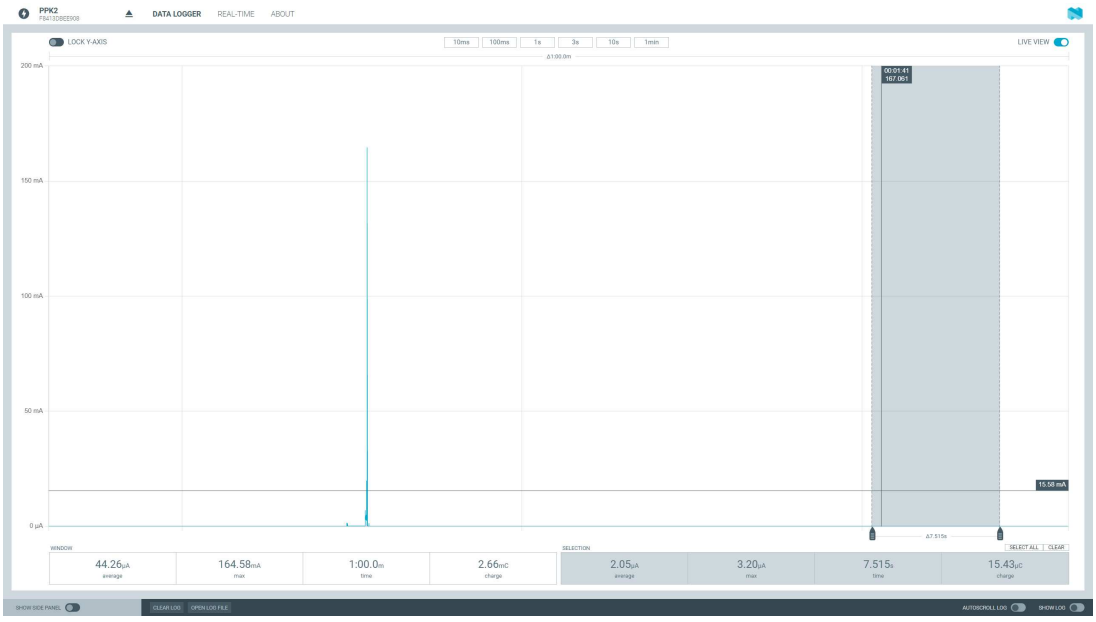

Figure 5.8. Power profile during sleep of the seventh version of the device with current consumption of 2 μA

6. Summary and conclusions

The original work described in this thesis includes implementation of firmware enabling ultra-low power mode of a microcontroller, communication with temperature sensor and radio module, design of an original energy harvesting circuit as well as design of PCBs used for performance measurements of both firmware and hardware. Due to disruption in supply chains in years 2021 and 2022 several prototypes with components that were available at that time were built. To store and visualise the acquired data a backend software was designed and implemented by the author of the thesis.

All the requirements regarding sensor listed in the first chapter have been meet. The produced device works as expected. The power consumption during sleep was even reduced down to 2 μA. All schematics, bills of materials, and source codes which could be useful in further development of energy-harvested wireless sensor network devices were provided in appendices.

A. Firmware source codes

Listing A.1. Main code for sensor and gateway with STM8L051 MCU

```
/**
  ******************************************************************************
  * @file    main.c
  * @author  Michal Markiewicz
  * @version V1.0.0
  * @date    21-October-2022
  * @brief   This file provides firmware functions for long range communication
  *          using SX127x LoRa module via SPI interface.
  *
  ******************************************************************************
  * @attention
  *
  * Communication every minute plus random number of seconds (<60).
  * LED always blinks during initalization and transmission.
  *
  * <h2><center>&copy; COPYRIGHT 2022 Michal Markiewicz</center></h2>
  *
  ******************************************************************************
  * @attention
  *
  * Pinout:
  *    GPIO  C4 [20] - VCC [SHT20]
  *     SDA  C0 [18] - SDA [SHT20]
  *     SCL  C1 [19] - SCL [SHT20]
  *
  *    MISO  B7 [17] - MISO [RA02]
  *    MOSI  B6 [16] - MOSI [RA02]
  *     SCK  B5 [15] - SCK  [RA02]
  *      CS  B4 [14] - NSS  [RA02]
  *    GPIO  B2 [12] - RST  [RA02]
  *    GPIO  B1 [11] - D0   [RA02]
  *
  * UART RX A3  [6]
  * UART TX A2  [5] - RPI
  *
  *  RESET PA1  [4] - STLINK
  *    SWIM PA0  [3] - STLINK
  *
  *    GPIO PB3 [13] - LED
```

```
40   *    GPIO PB0 [10] - DIP1 - 1 Hz with LED or 3mHz (5 min) without LED
41   *    GPIO PD0  [9]
42   *
43   *  OSC32 PC5  [1] - 32.768kHz
44   *  OSC32 PC6  [2] - 32.768kHz
45   *
46   ******************************************************************************
47   */
48
49 #define USE_FULL_ASSERT 1
50
51 #include "stm8l15x.h"
52 #include "voltage.h"
53 #include "sht2x.h"
54 #include "sx127x.h"
55 #include "stdio.h"
56 #include "aes.h"
57 #include "config.h"
58
59 #define TX
60 //#define RX
61 #define TLS 'A','T','N','E','R',' ','M','M',' ','-',' ','2','0','2','2','.'
62
63 /**
64   * @brief  LED
65   */
66 #define LED_GPIO_PORT                GPIOB
67 #define LED_PIN                      GPIO_Pin_3
68 #define LED_Init()                   GPIO_Init(LED_GPIO_PORT, LED_PIN,
      GPIO_Mode_Out_PP_High_Slow)
69 #define LED_ON()                     GPIO_ResetBits(LED_GPIO_PORT, LED_PIN)
70 #define LED_OFF()                    GPIO_SetBits(LED_GPIO_PORT, LED_PIN)
71
72
73 /**
74   * @brief  LED
75   */
76 #define SENSOR_POWER_GPIO_PORT       GPIOC
77 #define SENSOR_POWER_PIN             GPIO_Pin_4
78 #define SENSOR_POWER_Init()          GPIO_Init(SENSOR_POWER_GPIO_PORT, SENSOR_POWER_PIN,
      GPIO_Mode_Out_PP_High_Slow)
79 #define SENSOR_POWER_ON()            GPIO_ResetBits(SENSOR_POWER_GPIO_PORT, SENSOR_POWER_PIN)
80 #define SENSOR_POWER_OFF()           GPIO_SetBits(SENSOR_POWER_GPIO_PORT, SENSOR_POWER_PIN)
81
82
83 /**
84  * @brief Definition for UART COM port1
85  */
86 #define UART_COM_BAUD                115200
87 #define UART_COM                     USART1
88 #define UART_COM_GPIO_PORT           GPIOA
89 #define UART_COM_CLK                 CLK_Peripheral_USART1
90 #define UART_COM_RX_PIN              GPIO_Pin_3
91 #define UART_COM_TX_PIN              GPIO_Pin_2
```

```
92
93 /**
94   * @brief  DIP Switch DIP1 - power on/off
95   */
96 #define DIP2_GPIO_PORT                GPIOD
97 #define DIP2_PIN                      GPIO_Pin_0
98 #define DIP3_GPIO_PORT                GPIOB
99 #define DIP3_PIN                      GPIO_Pin_0
100
101
102 volatile uint16_t pulseCounter;
103
104 void delayus(uint16_t timeInUs)
105 {
106     while (timeInUs--)
107     {
108         asm("nop");
109         asm("nop");
110         asm("nop");
111         asm("nop");
112         asm("nop");
113         asm("nop");
114     }
115 }
116
117
118 /* Private functions ---------------------------------------------------------*/
119
120
121 #ifdef  USE_FULL_ASSERT
122
123 /**
124   * @brief  Reports the name of the source file and the source line number
125   *   where the assert_param error has occurred.
126   * @param  file: pointer to the source file name
127   * @param  line: assert_param error line source number
128   * @retval None
129   */
130 void assert_failed(uint8_t* file, uint32_t line)
131 {
132   /* User can add his own implementation to report the file name and line number,
133     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
134
135   /* Infinite loop */
136   while (1)
137   {
138   }
139 }
140 #endif
141
142
143 /**
144   * @brief  Configures UART COM port.
145   * @param  COM: Specifies the COM port to be configured.
```

```
146   *    This parameter should be COM1.
147   * @param  USART_InitStruct: pointer to a USART_InitTypeDef structure that
148   *    contains the configuration information for the specified USART peripheral.
149   * @retval None
150   */
151 void UART_Init()
152 {
153     CLK_PeripheralClockConfig(UART_COM_CLK, ENABLE);
154     GPIO_ExternalPullUpConfig(UART_COM_GPIO_PORT, UART_COM_TX_PIN | UART_COM_RX_PIN, ENABLE);
155     //GPIO_ExternalPullUpConfig(UART_COM_GPIO_PORT, UART_COM_RX_PIN, ENABLE);
156     if (UART_COM_GPIO_PORT == GPIOA)
157     {
158         SYSCFG_REMAPPinConfig(REMAP_Pin_USART1TxRxPortA, ENABLE);
159     }
160     USART_Init(UART_COM, (uint32_t)UART_COM_BAUD, USART_WordLength_8b,
161           USART_StopBits_1, USART_Parity_No,
162           (USART_Mode_TypeDef)(USART_Mode_Tx | USART_Mode_Rx));
163     USART_Cmd(UART_COM, ENABLE);
164 }
165
166 void UART_DeInit()
167 {
168     USART_Cmd(UART_COM, DISABLE);
169     CLK_PeripheralClockConfig(UART_COM_CLK, DISABLE);
170     GPIO_Init(UART_COM_GPIO_PORT, UART_COM_TX_PIN, GPIO_Mode_Out_PP_High_Slow);
171     GPIO_Init(UART_COM_GPIO_PORT, UART_COM_TX_PIN, GPIO_Mode_Out_PP_High_Slow);
172 }
173
174 void UART_TX(uint8_t c)
175 {
176     USART_SendData8(UART_COM, c);
177     while (USART_GetFlagStatus(UART_COM, USART_FLAG_TC) == RESET)
178     {
179     }
180 }
181
182 void delayms(uint16_t timeInMs)
183 {
184      while (timeInMs--)
185     {
186         for (uint16_t i = 380; i > 0; i--)
187         {
188             delayus(1);
189         }
190     }
191 }
192
193
194 /**
195   * @brief RTC / CSS_LSE Interrupt routine.
196   * @param  None
197   * @retval None
198   */
199 INTERRUPT_HANDLER(RTC_CSSLSE_IRQHandler, 4)
```

```
200 {
201 }
202
203 /**
204   * @brief External IT PORTD /PORTH Interrupt routine.
205   * @param  None
206   * @retval None
207   */
208 INTERRUPT_HANDLER(EXTID_H_IRQHandler, 7)
209 {
210     pulseCounter++;
211     EXTI_ClearITPendingBit(EXTI_IT_PortD);
212 }
213
214 void setUpPortDInterrupt()
215 {
216     EXTI_SelectPort(EXTI_Port_D);
217     EXTI_SetPortSensitivity(EXTI_Port_D, EXTI_Trigger_Falling);
218     EXTI_SetHalfPortSelection(EXTI_HalfPort_D_LSB, ENABLE);
219 }
220
221
222 void TurnOfPeripherals()
223 {
224     ADC_Cmd(ADC1, DISABLE);
225     BEEP_Cmd(DISABLE);
226     DAC_Cmd(DAC_Channel_1, DISABLE);
227     DAC_Cmd(DAC_Channel_2, DISABLE);
228     DMA_GlobalCmd(DISABLE);
229     I2C_Cmd(I2C1, DISABLE);
230     LCD_Cmd(DISABLE);
231     PWR_PVDCmd(DISABLE);
232     SPI_Cmd(SPI1, DISABLE);
233     TIM1_CtrlPWMOutputs(DISABLE);
234     TIM1_Cmd(DISABLE);
235     TIM2_CtrlPWMOutputs(DISABLE);
236     TIM2_Cmd(DISABLE);
237     TIM3_CtrlPWMOutputs(DISABLE);
238     TIM3_Cmd(DISABLE);
239     TIM4_Cmd(DISABLE);
240     USART_Cmd(USART1, DISABLE);
241     FLASH_PowerWaitModeConfig(FLASH_Power_IDDQ);
242     PWR_UltraLowPowerCmd(ENABLE);
243
244     GPIO_Init(GPIOA, GPIO_Pin_1, GPIO_Mode_Out_PP_High_Slow); //RESET
245     CFG->GCR |= CFG_GCR_SWD;
246     GPIO_Init(GPIOA, GPIO_Pin_0, GPIO_Mode_Out_PP_Low_Slow); //SWIM
247     GPIO_Init(GPIOA, GPIO_Pin_2 | GPIO_Pin_3, GPIO_Mode_Out_PP_High_Slow); //UART
248     GPIO_Init(LED_GPIO_PORT, LED_PIN, GPIO_Mode_Out_PP_High_Slow);
249     GPIO_Init(GPIOB, GPIO_Pin_0, GPIO_Mode_Out_PP_Low_Slow); //DIP1
250     GPIO_Init(GPIOD, GPIO_Pin_0, GPIO_Mode_Out_PP_Low_Slow); //DIP2
251     GPIO_Init(GPIOC, GPIO_Pin_0 | GPIO_Pin_1/*| GPIO_Pin_4*/, GPIO_Mode_Out_PP_High_Slow);
252     CLK_PeripheralClockConfig(CLK_Peripheral_TIM2, DISABLE);
253     CLK_PeripheralClockConfig(CLK_Peripheral_TIM3, DISABLE);
```

```
254     CLK_PeripheralClockConfig(CLK_Peripheral_TIM4, DISABLE);
255     CLK_PeripheralClockConfig(CLK_Peripheral_I2C1, DISABLE);
256     CLK_PeripheralClockConfig(CLK_Peripheral_SPI1, DISABLE);
257     CLK_PeripheralClockConfig(CLK_Peripheral_USART1, DISABLE);
258     CLK_PeripheralClockConfig(CLK_Peripheral_BEEP, DISABLE);
259     CLK_PeripheralClockConfig(CLK_Peripheral_DAC, DISABLE);
260     CLK_PeripheralClockConfig(CLK_Peripheral_ADC1, DISABLE);
261     CLK_PeripheralClockConfig(CLK_Peripheral_TIM1, DISABLE);
262     CLK_PeripheralClockConfig(CLK_Peripheral_LCD, DISABLE);
263     CLK_PeripheralClockConfig(CLK_Peripheral_DMA1, DISABLE);
264     CLK_PeripheralClockConfig(CLK_Peripheral_COMP, DISABLE);
265     CLK_PeripheralClockConfig(CLK_Peripheral_BOOTROM, DISABLE);
266     CLK_PeripheralClockConfig(CLK_Peripheral_AES, DISABLE);
267     CLK_PeripheralClockConfig(CLK_Peripheral_TIM5, DISABLE);
268     CLK_PeripheralClockConfig(CLK_Peripheral_SPI2, DISABLE);
269     CLK_PeripheralClockConfig(CLK_Peripheral_USART2, DISABLE);
270     CLK_PeripheralClockConfig(CLK_Peripheral_USART3, DISABLE);
271     CLK_PeripheralClockConfig(CLK_Peripheral_CSSLSE, DISABLE);
272 }
273
274 void RTC_SetUpWakeupTime(uint8_t min, uint8_t sec)
275 {
276     RTC_InitTypeDef   RTC_InitStr;
277     RTC_TimeTypeDef   RTC_TimeStr;
278     RTC_DateTypeDef   RTC_DateStr;
279     RTC_AlarmTypeDef  RTC_AlarmStr;
280
281     RTC_AlarmCmd(DISABLE);
282     RTC_ITConfig(RTC_IT_ALRA, DISABLE);
283
284     RTC_InitStr.RTC_HourFormat = RTC_HourFormat_24;
285     RTC_InitStr.RTC_AsynchPrediv = 0x7F;
286     RTC_InitStr.RTC_SynchPrediv = 0x00FF;
287
288     RTC_DateStructInit(&RTC_DateStr);
289     RTC_DateStr.RTC_WeekDay = RTC_Weekday_Friday;
290     RTC_DateStr.RTC_Date = 13;
291     RTC_DateStr.RTC_Month = RTC_Month_May;
292     RTC_DateStr.RTC_Year = 11;
293
294     RTC_TimeStructInit(&RTC_TimeStr);
295     RTC_TimeStr.RTC_Hours   = 01;
296     RTC_TimeStr.RTC_Minutes = 00;
297     RTC_TimeStr.RTC_Seconds = 00;
298
299     RTC_AlarmStructInit(&RTC_AlarmStr);
300     RTC_AlarmStr.RTC_AlarmTime.RTC_Hours   = 01;
301     RTC_AlarmStr.RTC_AlarmTime.RTC_Minutes = min;
302     RTC_AlarmStr.RTC_AlarmTime.RTC_Seconds = sec;
303     RTC_AlarmStr.RTC_AlarmMask = RTC_AlarmMask_DateWeekDay;
304
305     RTC_Init(&RTC_InitStr);
306     RTC_SetDate(RTC_Format_BIN, &RTC_DateStr);
307     RTC_SetTime(RTC_Format_BIN, &RTC_TimeStr);
```

```
308     RTC_SetAlarm(RTC_Format_BIN, &RTC_AlarmStr);
309     RTC_ITConfig(RTC_IT_ALRA, ENABLE);
310     RTC_AlarmCmd(ENABLE);
311 }
312
313 void GPIO_DIPStatus(uint8_t *dip2, uint8_t *dip3)
314 {
315     GPIO_ExternalPullUpConfig(DIP2_GPIO_PORT, DIP2_PIN, ENABLE);
316     GPIO_ExternalPullUpConfig(DIP3_GPIO_PORT, DIP3_PIN, ENABLE);
317     GPIO_Init(DIP2_GPIO_PORT, DIP2_PIN, GPIO_Mode_In_PU_No_IT); //DIP2
318     GPIO_Init(DIP3_GPIO_PORT, DIP3_PIN, GPIO_Mode_In_PU_No_IT); //DIP3
319     delayus(100);
320     *dip2 = GPIO_ReadInputDataBit(DIP2_GPIO_PORT, DIP2_PIN) == RESET;
321     *dip3 = GPIO_ReadInputDataBit(DIP3_GPIO_PORT, DIP3_PIN) == RESET;
322     GPIO_ExternalPullUpConfig(DIP2_GPIO_PORT, DIP2_PIN, DISABLE);
323     GPIO_ExternalPullUpConfig(DIP3_GPIO_PORT, DIP3_PIN, DISABLE);
324     GPIO_Init(DIP2_GPIO_PORT, DIP2_PIN, GPIO_Mode_Out_PP_Low_Slow);
325     GPIO_Init(DIP3_GPIO_PORT, DIP3_PIN, GPIO_Mode_Out_PP_Low_Slow);
326 }
327
328 #ifndef MAX
329 #define MAX(a, b) (a) > (b) ? (a) : (b)
330 #endif
331 #ifndef MIN
332 #define MIN(a, b) (a) < (b) ? (a) : (b)
333 #endif
334
335 #define MSB(x) (uint8_t)(((x) >> 8) & 0xFF)
336 #define LSB(x) (uint8_t)((x) & 0xFF);
337 #define H_NIB(x) (uint8_t)(((x) >> 4) & 0xF)
338 #define L_NIB(x) (uint8_t)((x) & (0xF))
339 #define NIB2TEXT(x) (((x) < 0xA) ? (x) + '0' : (x) - 0xA + 'A')
340
341 #define MSG_MAX_ID_LENGTH 8
342 #define PACKET_SIZE (1 + 1 + MSG_MAX_ID_LENGTH + 2 + 1 + 2 + 1 + 1)
343 #define CRC_POLYNOMIAL 0x131   //P(x)=x^8+x^5+x^4+1 = 100110001
344
345 uint8_t crc(uint8_t data[], uint8_t nbrOfBytes)
346 {
347     uint8_t crc = 0;
348     //calculates 8-Bit checksum with given polynomial
349     for (uint8_t  byteCtr = 0; byteCtr < nbrOfBytes; ++byteCtr)
350     {
351         crc ^= (data[byteCtr]);
352         for (uint8_t bit = 8; bit > 0; --bit)
353         {
354             if (crc & 0x80)
355             {
356                 crc = (crc << 1) ^ (uint16_t)(CRC_POLYNOMIAL);
357             }
358             else
359             {
360                 crc = (crc << 1);
361             }
```

```
362         }
363     }
364     return crc;
365 }
366
367 void encodePacket(uint8_t *buffer, uint8_t sn[], uint8_t snlen,
368     int16_t temperatureCX100, uint8_t humidity, uint16_t voltageX100, uint8_t sequenceNo)
369 {
370     uint8_t idx = 1;
371     buffer[idx++] = snlen;
372     for (uint8_t i = 0; i < snlen; i++)
373     {
374         buffer[idx++] = sn[i];
375     }
376     buffer[idx++] = MSB(temperatureCX100);
377     buffer[idx++] = LSB(temperatureCX100);
378     buffer[idx++] = humidity;
379     buffer[idx++] = MSB(voltageX100);
380     buffer[idx++] = LSB(voltageX100);
381     buffer[idx++] = sequenceNo;
382     buffer[idx++] = crc(buffer, PACKET_SIZE - 1);
383     buffer[0] = idx;
384     assert_param(idx == PACKET_SIZE);
385 }
386
387 void decodePacket(char *message, uint8_t msgMaxLength, uint8_t *packet, uint8_t packetLength,
    uint16_t rssi)
388 {
389     uint8_t error = 0;
390     uint8_t packetIdx = 1;
391     uint8_t nodeIdLen = packet[packetIdx++];
392     error |= (packetLength != packet[0] || nodeIdLen >= packetLength || nodeIdLen >
    MSG_MAX_ID_LENGTH);
393     error |= packet[PACKET_SIZE - 1] != crc(packet, PACKET_SIZE - 1);
394
395     if (error)
396     {
397         message[0] = 0;
398         return;
399     }
400     char nodeIdInHex[(MSG_MAX_ID_LENGTH << 1) + 1];
401     uint8_t nodeIdInHexIdx = 0;
402     for (uint8_t i = 0; i < nodeIdLen; i++)
403     {
404         nodeIdInHex[nodeIdInHexIdx++] =  NIB2TEXT(H_NIB(packet[packetIdx]));
405         nodeIdInHex[nodeIdInHexIdx++] =  NIB2TEXT(L_NIB(packet[packetIdx]));
406         packetIdx++;
407     }
408     nodeIdInHex[nodeIdInHexIdx] = 0;
409
410     int16_t tempCX100 = (int16_t)(packet[packetIdx++]) << 8;
411     tempCX100 += packet[packetIdx++];
412     uint8_t humidity = packet[packetIdx++];
413     uint16_t voltX100 = (int16_t)(packet[packetIdx++]) << 8;
```

```
414     voltX100 += packet[packetIdx++];
415     uint8_t sequenceNo = packet[packetIdx++];
416 #define FORMAT_GPS "$SENSOR,%s,SEQUENCE,%d,TEMPC,%d.%02d,HUMIDITY,%d,VOLTAGE,%d.%02d,RSSI,%d\r
        \n"
417 #define FORMAT_JSON "\"id\":\"%s\",\"sequence\":%d,\"temperature\":%d.%02d,\"humidity\":%d,\"
        voltage\":%d.%02d,\"rssi\":%d\r\n"
418 #define MESSAGE_FORMAT FORMAT_JSON
419     sprintf(message, MESSAGE_FORMAT,
420         nodeIdInHex,
421         sequenceNo,
422         tempCX100 / 100, (tempCX100 > 0 ? tempCX100 : -tempCX100) % 100,
423         humidity,
424         voltX100 / 100, voltX100 % 100,
425         rssi);
426 }
427
428 #ifdef TX
429 int main(void)
430 {
431     pulseCounter = 0;
432     SHT2x_TypeDef SHT20;
433     SX127x_TypeDef SX1278;
434     CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_1);
435     LED_Init();
436     SENSOR_POWER_Init();
437     LED_ON();
438     SENSOR_POWER_ON();
439     uint8_t sn[SHT2x_SERIAL_NUMBER_LENGTH];
440     SHT2x_Init();
441     SHT2x_Error SHT2x_err = SHT2x_NO_ERROR;
442     do {
443         SHT2x_err = SHT2x_SoftReset();
444         SHT2x_err |= SHT2x_GetSerialNumber(sn);
445     } while (SHT2x_err != SHT2x_NO_ERROR);
446     CLK_HSEConfig(CLK_HSE_OFF);
447     CLK_LSICmd(DISABLE);
448     CLK_LSEConfig(CLK_LSE_ON);
449     CLK_RTCClockConfig(CLK_RTCCLKSource_LSE, CLK_RTCCLKDiv_1);
450     CLK_PeripheralClockConfig(CLK_Peripheral_RTC, ENABLE);
451     while (RTC_WaitForSynchro() != SUCCESS);
452     enableInterrupts();
453     LED_OFF();
454     uint8_t buffer[PACKET_SIZE + 1];
455     uint8_t sequenceNo = 0;
456 #ifdef TLS
457     static const uint8_t key[] = {TLS};
458     struct AES_ctx ctx;
459     AES_init_ctx(&ctx, key);
460 #endif
461     while (1)
462     {
463         CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_1);
464         //HSI
465         CLK_HSICmd(ENABLE);
```

```
466         CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_HSI);
467         CLK_SYSCLKSourceSwitchCmd(ENABLE);
468         while (CLK_GetFlagStatus(CLK_FLAG_HSIRDY) == 0);
469         SENSOR_POWER_ON();
470         do {
471             SHT2x_err = SHT2x_SoftReset();
472             SHT2x_err |= SHT2x_MeasurePoll(&SHT20, TEMPERATURE);
473             SHT2x_err |= SHT2x_MeasurePoll(&SHT20, HUMIDITY);
474         } while (SHT2x_err != SHT2x_NO_ERROR);
475         uint16_t voltageX100 = Voltage_Measure_ResultIn10mV();
476         encodePacket(buffer, sn, SHT2x_SERIAL_NUMBER_LENGTH,
477             SHT20.temperatureCX100, SHT20.humidity, voltageX100, sequenceNo++);
478 #ifdef TLS
479         AES_ECB_encrypt(&ctx, buffer);
480 #endif
481         LED_ON();
482         while (SX127x_Init(&SX1278) != SX127x_NO_ERROR)
483         {
484         }
485         SX127x_BeginPacket(&SX1278);
486         SX127x_Append(buffer, PACKET_SIZE);
487         SX127x_EndPacket();
488         //uint8_t randomByte = SX127x_Random(&SX1278);
489         SX127x_Sleep();
490         SX127x_DeInit();
491         LED_OFF();
492         TurnOfPeripherals();
493         SENSOR_POWER_OFF();
494         uint16_t randomByte = (uint16_t)(voltageX100 * 10000);
495         RTC_SetUpWakeupTime(1, (randomByte) % 60);
496         //LSE
497         CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_LSE);
498         CLK_SYSCLKSourceSwitchCmd(ENABLE);
499         while (CLK_GetFlagStatus(CLK_FLAG_LSERDY) == 0);
500         CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_128);
501         CLK_HSICmd(DISABLE);
502
503         RTC_ClearITPendingBit(RTC_IT_ALRA);
504         RTC_ClearITPendingBit(RTC_IT_WUT);
505         while (RTC_GetITStatus(RTC_IT_ALRA) == RESET &&
506                RTC_GetITStatus(RTC_IT_WUT) == RESET)
507         {
508             halt();
509         }
510
511     }
512 }
513 #endif
514
515 #ifdef RX
516 int main(void)
517 {
518     CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_1);
519     SX127x_TypeDef SX1278;
```

```
520
521     LED_Init();
522     LED_ON();
523     delayms(200);
524     while (SX127x_Init(&SX1278) != SX127x_NO_ERROR)
525     {
526     }
527     LED_OFF();
528
529     SX127x_Standby(&SX1278);
530     UART_Init();
531 #ifdef TLS
532     const uint8_t key[] = {TLS};
533     struct AES_ctx ctx;
534     AES_init_ctx(&ctx, key);
535 #endif
536      uint8_t packet[PACKET_SIZE + 1];
537 #define MAX_MESSAGE_SIZE 128
538     char message[MAX_MESSAGE_SIZE];
539     while (1)
540     {
541         if (SX127x_ParsePacket(&SX1278, PACKET_SIZE))
542         {
543             LED_ON();
544             uint8_t pIdx = 0;
545             while (SX127x_Available(&SX1278) && pIdx < PACKET_SIZE)
546             {
547                 uint16_t c = SX127x_ReadPacket(&SX1278);
548                 if (c != SX127X_EOF)
549                 {
550                     packet[pIdx++] = (uint8_t)c;
551                 }
552             }
553             if (pIdx != PACKET_SIZE)
554             {
555                 continue;
556             }
557 #ifdef TLS
558             AES_ECB_decrypt(&ctx, packet);
559 #endif
560             int16_t rssi = SX127x_PacketRssi(&SX1278);
561             decodePacket(message, MAX_MESSAGE_SIZE, packet, PACKET_SIZE, rssi);
562             char *p = message;
563             while (*p)
564             {
565                 UART_TX(*p++);
566             }
567             LED_OFF();
568         }
569     }
570 }
571 #endif
```

B. Backend source codes

B.1. Database script

Listing B.1. SQL Database creation script

```
CREATE TABLE t_buildings (
 bui_id INT IDENTITY (1,1) PRIMARY KEY,
 bui_street VARCHAR (50),
 bui_number VARCHAR (10),
 bui_city VARCHAR (50),
 bui_zip_code VARCHAR (6),
 bui_country VARCHAR (50),
 bui_lat FLOAT,
 bui_lon FLOAT,
 bui_owner VARCHAR(200),
 bui_building_type_dic VARCHAR(30)
);
GO

CREATE TABLE t_flats (
 fla_id INT IDENTITY (1,1) PRIMARY KEY,
 fla_bui_id INT REFERENCES t_buildings(bui_id) NOT NULL,
 fla_number VARCHAR(10),
 fla_owner VARCHAR(50),
 fla_area_in_m2 FLOAT,
 fla_floor INT,
 fla_flat_type_dic VARCHAR(30)
);
GO

CREATE TABLE t_rooms (
 roo_id INT IDENTITY (1,1) PRIMARY KEY,
 roo_fla_id INT REFERENCES t_flats(fla_id),
 roo_name VARCHAR(200),
 roo_area_in_m2 FLOAT,
 roo_windows_direction_dic VARCHAR(30),
 roo_room_type_dic VARCHAR(30)
);
GO


```

```
37 CREATE TABLE t_sensors (
38  sen_id INT IDENTITY (1,1) PRIMARY KEY,
39  sen_type_dic VARCHAR(20),
40  sen_name VARCHAR(200)
41 );
42 GO
43
44 CREATE TABLE t_installed_sensors (
45  ins_id INT IDENTITY (1,1) PRIMARY KEY,
46  ins_serial_number VARCHAR(50) UNIQUE,
47  ins_sen_id INT NULL REFERENCES t_sensors(sen_id),
48  ins_roo_id INT NULL REFERENCES t_rooms(roo_id),
49  ins_audit_cd DATETIME DEFAULT GETDATE(),
50  ins_name VARCHAR(200),
51  ins_access_key VARCHAR(50)
52 );
53 GO
54
55 CREATE TABLE t_readouts (
56  rea_id INT IDENTITY (1,1) PRIMARY KEY,
57  rea_ins_id INT REFERENCES t_installed_sensors(ins_id),
58  rea_audit_cd DATETIME DEFAULT GETDATE(),
59  rea_data FLOAT,
60  rea_sequence TINYINT, /* 0-255 */
61  rea_temperature FLOAT(24), /* 4 bytes */
62  rea_humidity FLOAT(24), /* 4 bytes */
63  rea_rssi SMALLINT, /* 2 bytes with sign */
64  rea_voltage FLOAT(24), /* 4 bytes */
65  rea_timestamp DATETIME
66 );
67 GO
68
69 CREATE TABLE t_dictionaries (
70  dic_id INT IDENTITY (1,1) PRIMARY KEY,
71  dic_group VARCHAR(30),
72  dic_key VARCHAR(30),
73  dic_value VARCHAR(200)
74 );
75 GO
76
77 CREATE VIEW v_sensors AS
78 SELECT rea_audit_cd, rea_data, ins_name, sen_type_dic, sen_name, roo_room_type_dic, bui_street
       , bui_city
79 FROM t_readouts
80 JOIN t_installed_sensors on ins_id = rea_ins_id
81 JOIN t_sensors on sen_id = ins_sen_id
82 JOIN t_rooms on roo_id = ins_roo_id
83 JOIN t_flats on fla_id = roo_fla_id
84 JOIN t_buildings on bui_id = fla_bui_id;
85 GO
86
87 CREATE TABLE t_parameters (
88  par_id INT IDENTITY (1,1) PRIMARY KEY,
89  par_name VARCHAR(50),
```

```
 par_value VARCHAR(200),
 par_description VARCHAR(200)
);
GO


----------------------------------------
-- Buildings
----------------------------------------

DROP PROCEDURE p_buildings_select_all;
GO

CREATE PROCEDURE p_buildings_select_all AS
BEGIN
 SELECT * FROM T_BUILDINGS
END
GO

DROP PROCEDURE p_buildings_select_single;
GO
CREATE PROCEDURE p_buildings_select_single (@bui_id int) AS
BEGIN
  SELECT * FROM T_BUILDINGS WHERE bui_id = @bui_id
END
GO

DROP PROCEDURE p_buildings_delete;
GO

CREATE PROCEDURE p_buildings_delete (@bui_id int) AS
BEGIN
  DELETE t_buildings WHERE bui_id = @bui_id
END
GO

DROP PROCEDURE p_buildings_insert_update;
GO

CREATE PROCEDURE p_buildings_insert_update (
@bui_id int,
@bui_street VARCHAR (50) = null,
@bui_number VARCHAR (10) = null,
@bui_city VARCHAR (50) = null,
@bui_zip_code VARCHAR (6) = null,
@bui_country VARCHAR (50) = null,
@bui_lat FLOAT = null,
@bui_lon FLOAT = null,
@bui_owner VARCHAR(200) = null,
@bui_building_type_dic VARCHAR(20) = null)
AS
BEGIN
 IF @bui_id = 0
```

```
143   INSERT INTO t_buildings(bui_street, bui_number, bui_city, bui_zip_code, bui_country, bui_lat
        , bui_lon, bui_owner, bui_building_type_dic)
144   VALUES (@bui_street, @bui_number, @bui_city, @bui_zip_code, @bui_country, @bui_lat, @bui_lon
        , @bui_owner, @bui_building_type_dic)
145  ELSE
146   UPDATE t_buildings
147   SET
148     bui_street = @bui_street,
149     bui_number = @bui_number,
150     bui_city = @bui_city,
151     bui_zip_code =@bui_zip_code,
152     bui_country =@bui_country,
153     bui_lat = @bui_lat,
154     bui_lon = @bui_lon,
155     bui_owner =@bui_owner,
156     bui_building_type_dic =@bui_building_type_dic
157   WHERE bui_id = @bui_id
158 END
159 GO
160
161 ---------------------------------------
162 -- Flats
163 ---------------------------------------
164
165 DROP PROCEDURE p_flats_select_all;
166 GO
167
168 CREATE PROCEDURE p_flats_select_all AS
169 BEGIN
170  SELECT * FROM T_FLATS
171 END
172 GO
173
174 DROP PROCEDURE p_flats_select_single;
175 GO
176 CREATE PROCEDURE p_flats_select_single (@fla_id int) AS
177 BEGIN
178   SELECT * FROM T_FLATS WHERE fla_id = @fla_id
179 END
180 GO
181
182 DROP PROCEDURE p_flats_delete;
183 GO
184
185 CREATE PROCEDURE p_flats_delete (@fla_id int) AS
186 BEGIN
187   DELETE t_flats WHERE fla_id = @fla_id
188 END
189 GO
190
191 DROP PROCEDURE p_flats_insert_update;
192 GO
193
194 CREATE PROCEDURE p_flats_insert_update (
```

```
195 @fla_id int,
196 @fla_bui_id int,
197 @fla_number VARCHAR (10) = null,
198 @fla_owner VARCHAR (50) = null,
199 @fla_area_in_m2 FLOAT = null,
200 @fla_floor INT,
201 @fla_flat_type_dic VARCHAR(20) = null)
202
203
204 AS
205 BEGIN
206  IF @fla_id = 0
207   INSERT INTO t_flats(fla_bui_id, fla_number, fla_owner, fla_area_in_m2, fla_floor,
        fla_flat_type_dic)
208   VALUES (@fla_bui_id, @fla_number, @fla_owner, @fla_area_in_m2, @fla_floor,
        @fla_flat_type_dic)
209  ELSE
210   UPDATE t_flats
211   SET
212     fla_bui_id = @fla_bui_id,
213     fla_number = @fla_number,
214     fla_owner = @fla_owner,
215     fla_area_in_m2 = @fla_area_in_m2,
216     fla_floor = @fla_floor,
217     fla_flat_type_dic = @fla_flat_type_dic
218   WHERE fla_id = @fla_id
219 END
220 GO
221
222 ---------------------------------------
223 -- Rooms
224 ---------------------------------------
225
226 DROP PROCEDURE p_rooms_select_all;
227 GO
228
229 CREATE PROCEDURE p_rooms_select_all AS
230 BEGIN
231  SELECT * FROM T_ROOMS
232 END
233 GO
234
235 DROP PROCEDURE p_rooms_select_single;
236 GO
237 CREATE PROCEDURE p_rooms_select_single (@roo_id int) AS
238 BEGIN
239   SELECT * FROM T_ROOMS WHERE roo_id = @roo_id
240 END
241 GO
242
243 DROP PROCEDURE p_rooms_delete;
244 GO
245
246 CREATE PROCEDURE p_rooms_delete (@roo_id int) AS
```

```
247 BEGIN
248   DELETE t_rooms WHERE roo_id = @roo_id
249 END
250 GO
251
252 DROP PROCEDURE p_rooms_insert_update;
253 GO
254
255 CREATE PROCEDURE p_rooms_insert_update (
256 @roo_id int,
257 @roo_fla_id int,
258 @roo_name VARCHAR (200) = null,
259 @roo_area_in_m2 FLOAT = null,
260 @roo_windows_direction_dic VARCHAR(20) = null,
261 @roo_room_type_dic VARCHAR(20) = null)
262
263
264 AS
265 BEGIN
266  IF @roo_id = 0
267   INSERT INTO t_rooms(roo_fla_id, roo_name, roo_area_in_m2, roo_windows_direction_dic,
       roo_room_type_dic)
268   VALUES (@roo_fla_id, @roo_name, @roo_area_in_m2, @roo_windows_direction_dic,
       @roo_room_type_dic)
269  ELSE
270   UPDATE t_rooms
271   SET
272     roo_fla_id = @roo_fla_id,
273     roo_name = @roo_name,
274     roo_area_in_m2 = @roo_area_in_m2,
275     roo_windows_direction_dic = @roo_windows_direction_dic,
276     roo_room_type_dic = @roo_room_type_dic
277   WHERE roo_id = @roo_id
278 END
279 GO
280
281 ---------------------------------------
282 -- Sensors
283 ---------------------------------------
284
285 DROP PROCEDURE p_sensors_select_all;
286 GO
287
288 CREATE PROCEDURE p_sensors_select_all AS
289 BEGIN
290  SELECT * FROM T_SENSORS
291 END
292 GO
293
294 DROP PROCEDURE p_sensors_select_single;
295 GO
296 CREATE PROCEDURE p_sensors_select_single (@sen_id int) AS
297 BEGIN
298   SELECT * FROM T_SENSORS WHERE sen_id = @sen_id
```

```
299 END
300 GO
301
302 DROP PROCEDURE p_sensors_delete;
303 GO
304
305 CREATE PROCEDURE p_sensors_delete (@sen_id int) AS
306 BEGIN
307   DELETE t_sensors WHERE sen_id = @sen_id
308 END
309 GO
310
311 DROP PROCEDURE p_sensors_insert_update;
312 GO
313
314 CREATE PROCEDURE p_sensors_insert_update (
315 @sen_id int,
316 @sen_type_dic VARCHAR (20) = null,
317 @sen_name VARCHAR(200) = null)
318
319
320 AS
321 BEGIN
322  IF @sen_id = 0
323   INSERT INTO t_sensors(sen_type_dic, sen_name)
324   VALUES (@sen_type_dic, @sen_name)
325  ELSE
326   UPDATE t_sensors
327   SET
328     sen_type_dic = @sen_type_dic,
329     sen_name = @sen_name
330   WHERE sen_id = @sen_id
331 END
332 GO
333
334 ---------------------------------------
335 -- Installed_sensors
336 ---------------------------------------
337
338 DROP PROCEDURE p_installed_sensors_select_all;
339 GO
340
341 CREATE PROCEDURE p_installed_sensors_select_all AS
342 BEGIN
343  SELECT * FROM T_INSTALLED_SENSORS
344 END
345 GO
346
347 DROP PROCEDURE p_installed_sensors_select_single;
348 GO
349 CREATE PROCEDURE p_installed_sensors_select_single (@ins_id int) AS
350 BEGIN
351   SELECT * FROM T_INSTALLED_SENSORS WHERE ins_id = @ins_id
352 END
```

```
353 GO
354
355 DROP PROCEDURE p_installed_sensors_delete;
356 GO
357
358 CREATE PROCEDURE p_installed_sensors_delete (@ins_id int) AS
359 BEGIN
360   DELETE t_installed_sensors WHERE ins_id = @ins_id
361 END
362 GO
363
364 DROP PROCEDURE p_installed_sensors_insert_update;
365 GO
366
367 CREATE PROCEDURE p_installed_sensors_insert_update (
368 @ins_id int,
369 @ins_sen_id int = null,
370 @ins_roo_id int = null,
371 @ins_name VARCHAR(200) = null,
372 @ins_serial_number VARCHAR(50),
373 @ins_access_key VARCHAR(50) = null)
374
375
376 AS
377 BEGIN
378  IF @ins_id = 0
379   INSERT INTO t_installed_sensors(ins_sen_id, ins_roo_id, ins_name, ins_serial_number,
       ins_access_key)
380   VALUES (@ins_sen_id, @ins_roo_id, @ins_name, @ins_serial_number, @ins_access_key)
381  ELSE
382   UPDATE t_installed_sensors
383   SET
384     ins_sen_id = @ins_sen_id,
385     ins_roo_id = @ins_roo_id,
386     ins_name = @ins_name,
387     ins_serial_number = @ins_serial_number,
388     ins_access_key = @ins_access_key
389   WHERE ins_id = @ins_id
390 END
391 GO
392
393 ----------------------------------------
394 -- Readouts
395 ----------------------------------------
396
397 DROP PROCEDURE p_readouts_select_all;
398 GO
399
400 CREATE PROCEDURE p_readouts_select_all AS
401 BEGIN
402  SELECT TOP 100 * FROM T_READOUTS ORDER BY rea_audit_cd DESC
403 END
404 GO
405
```

```
406 DROP PROCEDURE p_readouts_select_single;
407 GO
408 CREATE PROCEDURE p_readouts_select_single (@rea_id int) AS
409 BEGIN
410   SELECT * FROM T_READOUTS WHERE rea_id = @rea_id
411 END
412 GO
413
414 DROP PROCEDURE p_readouts_delete;
415 GO
416
417 CREATE PROCEDURE p_readouts_delete (@rea_id int) AS
418 BEGIN
419   DELETE t_readouts WHERE rea_id = @rea_id
420 END
421 GO
422
423 DROP PROCEDURE p_readouts_insert_update;
424 GO
425
426 CREATE PROCEDURE p_readouts_insert_update (
427 @rea_id int,
428 @rea_ins_id int,
429 @rea_data FLOAT = null,
430 @rea_sequence TINYINT = null,
431 @rea_temperature FLOAT = null,
432 @rea_humidity FLOAT = null,
433 @rea_rssi SMALLINT = null,
434 @rea_voltage FLOAT = null,
435 @rea_timestamp DATETIME = null)
436 AS
437 BEGIN
438  IF @rea_id = 0
439     INSERT INTO t_readouts(rea_ins_id, rea_data, rea_sequence, rea_temperature, rea_humidity,
       rea_rssi, rea_voltage, rea_timestamp)
440     VALUES (@rea_ins_id, @rea_data, @rea_sequence, @rea_temperature, @rea_humidity, @rea_rssi,
        @rea_voltage, @rea_timestamp)
441   ELSE
442     UPDATE t_readouts
443     SET
444       rea_ins_id = @rea_ins_id,
445       rea_data = @rea_data,
446       rea_sequence = @rea_sequence,
447       rea_temperature = @rea_temperature,
448       rea_humidity = @rea_humidity,
449       rea_rssi = @rea_rssi,
450       rea_voltage = @rea_voltage,
451       rea_timestamp = @rea_timestamp
452     WHERE rea_id = @rea_id
453 END
454 GO
455
456 ---------------------------------------
457 -- Dictionaries
```

```
458 ---------------------------------------
459
460 DROP PROCEDURE p_dictionaries_select_all;
461 GO
462
463 CREATE PROCEDURE p_dictionaries_select_all AS
464 BEGIN
465   SELECT * FROM T_DICTIONARIES ORDER BY dic_group, dic_key
466 END
467 GO
468
469 DROP PROCEDURE p_dictionaries_select_single;
470 GO
471 CREATE PROCEDURE p_dictionaries_select_single (@dic_id int) AS
472
473 BEGIN
474    SELECT * FROM T_DICTIONARIES WHERE dic_id = @dic_id
475 END
476 GO
477
478 DROP PROCEDURE p_dictionaries_delete;
479 GO
480
481 CREATE PROCEDURE p_dictionaries_delete (@dic_id int) AS
482 BEGIN
483    DELETE t_dictionaries WHERE dic_id = @dic_id
484 END
485 GO
486
487 DROP PROCEDURE p_dictionaries_insert_update;
488 GO
489
490 CREATE PROCEDURE p_dictionaries_insert_update (
491 @dic_id INT,
492 @dic_group VARCHAR (30) = null,
493 @dic_key VARCHAR (30) = null,
494 @dic_value VARCHAR (200) = null)
495 AS
496 BEGIN
497      UPDATE t_dictionaries
498      SET
499         dic_group = @dic_group,
500         dic_key = @dic_key,
501         dic_value = @dic_value
502      WHERE dic_id = @dic_id
503
504 END
505 GO
506
507 ---------------------------------------
508 -- Parameters
509 ---------------------------------------
510
511 DROP PROCEDURE p_parameters_select_all;
```

```
512 GO
513
514 CREATE PROCEDURE p_parameters_select_all AS
515 BEGIN
516  SELECT * FROM T_PARAMETERS
517 END
518 GO
519
520 DROP PROCEDURE p_parameters_select_single;
521 GO
522 CREATE PROCEDURE p_parameters_select_single (@par_id int) AS
523 BEGIN
524   SELECT * FROM T_PARAMETERS WHERE par_id = @par_id
525 END
526 GO
527
528 DROP PROCEDURE p_parameters_delete;
529 GO
530
531 CREATE PROCEDURE p_parameters_delete (@par_id int) AS
532 BEGIN
533   DELETE t_parameters WHERE par_id = @par_id
534 END
535 GO
536
537 DROP PROCEDURE p_parameters_insert_update;
538 GO
539
540 CREATE PROCEDURE p_parameters_insert_update (
541 @par_id int,
542 @par_name VARCHAR (50) = null,
543 @par_value VARCHAR (200) = null,
544 @par_description VARCHAR (200) = null)
545 AS
546 BEGIN
547  IF @par_id = 0
548     INSERT INTO t_parameters(par_name, par_value, par_description)
549     VALUES (@par_name, @par_value, @par_description)
550   ELSE
551     UPDATE t_parameters
552     SET
553       par_name = @par_name,
554       par_value = @par_value,
555       par_description = @par_description
556     WHERE par_id = @par_id
557 END
558 GO
559
560
561 DROP PROCEDURE p_sensor_data_select;
562 GO
563
564 CREATE PROCEDURE p_sensor_data_select (
565 @rea_date_from  datetime = null,
```

```
@rea_date_to  datetime = null)
AS
BEGIN
 select bui_id, fla_id, roo_id, ins_id, sen_id, t_readouts.*, sen_type_dic
 from t_readouts
 join t_installed_sensors on (rea_ins_id = ins_id)
 join t_sensors on (sen_id = ins_sen_id)
 join t_rooms on (ins_roo_id = roo_id)
 join t_flats on (roo_fla_id = fla_id)
 join t_buildings on (bui_id = fla_bui_id)
 WHERE (@rea_date_from is null OR @rea_date_from <= rea_audit_cd)
   AND (@rea_date_to is null OR rea_audit_cd <= @rea_date_to)
 order by bui_id, fla_id, roo_id, ins_id, sen_id, rea_id
END
GO


---------------------------------
INSERT INTO t_buildings(bui_street, bui_number, bui_city, bui_zip_code, bui_country, bui_lat,
    bui_lon, bui_owner, bui_building_type_dic)
VALUES ('Adama Mickiewicza', '30', 'Cracow', '30-059', 'Polska', 50.03523, 19.55248, 'Akademia
     Gorniczo-Hutnicza', 'PUBLIC_BUILDING');

INSERT INTO t_flats(fla_bui_id, fla_number, fla_owner, fla_area_in_m2, fla_floor,
    fla_flat_type_dic)
VALUES (1, 1, 'Atner Sp. z o.o.', 20, 1, 'OFFICE');

INSERT INTO t_flats(fla_bui_id, fla_number, fla_owner, fla_area_in_m2, fla_floor,
    fla_flat_type_dic)
VALUES (1, 5, 'Atner Sp. z o.o.', 15, 2, 'GARAGE');

INSERT INTO t_rooms(roo_fla_id, roo_name, roo_area_in_m2, roo_windows_direction_dic,
    roo_room_type_dic)
VALUES (1, 'Kitchen', 10, 'W','KITCHEN');

INSERT INTO t_rooms(roo_fla_id, roo_name, roo_area_in_m2, roo_windows_direction_dic,
    roo_room_type_dic)
VALUES (1, 'Conference room', 20, 'S','CONFERENCE_ROOM');

INSERT INTO t_sensors(sen_type_dic, sen_name)
VALUES ('ATNITHB','Indoor temperature and humidity battery powered sensor');

INSERT INTO t_installed_sensors(ins_sen_id, ins_roo_id,ins_name, ins_serial_number,
    ins_access_key)
VALUES (1, 3, 'Temperature sensor in the conference room', 'SN-TEST1', '123');

INSERT INTO t_installed_sensors(ins_sen_id, ins_roo_id,ins_name, ins_serial_number,
    ins_access_key)
VALUES (1, 2, 'Temperature sensor in the kitchen', 'SN-TEST2', '123');

INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
VALUES ('BUILDING_TYPE', 'BLOCK_OF_FLATS', 'Block of flats');

INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
```

```
612 VALUES ('BUILDING_TYPE', 'PUBLIC_BUILDING', 'Public building');
613
614 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
615 VALUES ('FLAT_TYPE', 'OFFICE', 'Office');
616
617 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
618 VALUES ('FLAT_TYPE', 'FLAT', 'Flat');
619
620 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
621 VALUES ('FLAT_TYPE', 'GARAGE', 'Garage');
622
623 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
624 VALUES ('FLAT_TYPE', 'HEAT_NODE', 'Heat node');
625
626 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
627 VALUES ('ROOM_TYPE', 'CONFERENCE_ROOM', 'Conference room');
628
629 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
630 VALUES ('ROOM_TYPE', 'TOILET', 'Toilet');
631
632 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
633 VALUES ('ROOM_TYPE', 'KITCHEN', 'Kitchen');
634
635 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
636 VALUES ('ROOM_TYPE', 'LIVING_ROOM', 'Living room');
637
638 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
639 VALUES ('ROOM_TYPE', 'BEDROOM', 'Bedroom');
640
641 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
642 VALUES ('ROOM_TYPE', 'CORRIDOR', 'Corridor');
643
644 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
645 VALUES ('ROOM_TYPE', 'BATHROOM', 'Bathroom');
646
647 INSERT INTO t_dictionaries(dic_group, dic_key,dic_value)
648 VALUES ('ROOM_TYPE', 'TECHNICAL_ROOM', 'Technical room');
649
650 INSERT INTO t_parameters(par_name, par_value, par_description)
651 VALUES ('READOUTS_SERVER_IP', '*', 'IP addresses of all servers which are allowed to publish
      sensor data (comma separated).');
652
653 INSERT INTO t_parameters(par_name, par_value, par_description)
654 VALUES ('SAVE_READOUTS_TO_DB', '1', 'Save readouts received from sensors to the database (0/1)
      ');
655
656 INSERT INTO t_parameters(par_name, par_value, par_description)
657 VALUES ('READOUTS_QUEUE_LENGTH', '50', 'Readouts queue length for each installed sensor (>0)')
      ;
```

B.2. Data access service

Listing B.2. Data access service class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using DataCloud.Models;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System.Data;
using System.Data.SqlClient;
using System.IO;

namespace DataCloud.Data
{
    public enum ErrorCode
    {
        OK = 0,
        ERROR = 1,
    }

    public class DataAccessService : IDisposable
    {
        private readonly SqlConnection sqlConnection;

        public DataAccessService(IConfiguration configuration)
        {
            sqlConnection = new SqlConnection(configuration.GetConnectionString("
    DefaultConnection"));
            sqlConnection.Open();
            sqlConnection.Close();
        }


        public IEnumerable<TBuildings> TBuildingsSelectAll()
        {
            List<TBuildings> items = new List<TBuildings>();
            using (SqlCommand cmd = new SqlCommand("p_buildings_select_all", sqlConnection))
            {
                cmd.CommandType = CommandType.StoredProcedure;
                try
                {
                    sqlConnection.Open();
                    SqlDataReader rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        TBuildings item = new TBuildings();
                        item.BuiId = rdr.GetInt32("bui_id");
                        item.BuiStreet = rdr["bui_street"].ToString();
                        item.BuiNumber = rdr["bui_number"].ToString();
                        item.BuiCity = rdr["bui_city"].ToString();
                        item.BuiZipCode = rdr["bui_zip_code"].ToString();
                        item.BuiCountry = rdr["bui_country"].ToString();
                        item.BuiLat = ExtractDoubleValue(rdr, "bui_lat");
                        item.BuiLon = ExtractDoubleValue(rdr, "bui_lon");
```

```
54                          item.BuiOwner = rdr["bui_owner"].ToString();
55                          item.BuiBuildingTypeDic = rdr["bui_building_type_dic"].ToString();
56                          items.Add(item);
57                      }
58                  }
59                  finally
60                  {
61                      sqlConnection.Close();
62                  }
63              }
64              return items;
65          }
66
67          public TBuildings TBuildingsSelectSingle(int id)
68          {
69              TBuildings item = new TBuildings();
70              using (SqlCommand cmd = new SqlCommand("p_buildings_select_single", sqlConnection)
    )
71              {
72                  cmd.CommandType = CommandType.StoredProcedure;
73                  cmd.Parameters.AddWithValue("@bui_id", id);
74                  try
75                  {
76                      sqlConnection.Open();
77                      SqlDataReader rdr = cmd.ExecuteReader();
78                      if (rdr.Read())
79                      {
80                          item.BuiId = rdr.GetInt32("bui_id");
81                          item.BuiStreet = rdr["bui_street"].ToString();
82                          item.BuiNumber = rdr["bui_number"].ToString();
83                          item.BuiCity = rdr["bui_city"].ToString();
84                          item.BuiZipCode = rdr["bui_zip_code"].ToString();
85                          item.BuiCountry = rdr["bui_country"].ToString();
86                          item.BuiLat = ExtractDoubleValue(rdr, "bui_lat");
87                          item.BuiLon = ExtractDoubleValue(rdr, "bui_lon");
88                          item.BuiOwner = rdr["bui_owner"].ToString();
89                          item.BuiBuildingTypeDic = rdr["bui_building_type_dic"].ToString();
90                      }
91                  }
92                  finally
93                  {
94                      sqlConnection.Close();
95                  }
96              }
97              return item;
98          }
99
100         public void TBuildingsInsertUpdate(TBuildings item)
101         {
102             using (SqlCommand cmd = new SqlCommand("p_buildings_insert_update", sqlConnection)
    )
103             {
104                 cmd.CommandType = CommandType.StoredProcedure;
105                 cmd.Parameters.AddWithValue("@bui_id", item.BuiId);
```

```
106                 cmd.Parameters.AddWithValue("@bui_street", item.BuiStreet);
107                 cmd.Parameters.AddWithValue("@bui_number", item.BuiNumber);
108                 cmd.Parameters.AddWithValue("@bui_city", item.BuiCity);
109                 cmd.Parameters.AddWithValue("@bui_zip_code", item.BuiZipCode);
110                 cmd.Parameters.AddWithValue("@bui_country", item.BuiCountry);
111                 cmd.Parameters.AddWithValue("@bui_lat", item.BuiLat);
112                 cmd.Parameters.AddWithValue("@bui_lon", item.BuiLon);
113                 cmd.Parameters.AddWithValue("@bui_owner", item.BuiOwner);
114                 cmd.Parameters.AddWithValue("@bui_building_type_dic", item.BuiBuildingTypeDic)
    ;
115                 try
116                 {
117                     sqlConnection.Open();
118                     cmd.ExecuteNonQuery();
119                 }
120                 finally
121                 {
122                     sqlConnection.Close();
123                 }
124             }
125         }
126         public void TBuildingsDelete(int id)
127         {
128             using (SqlCommand cmd = new SqlCommand("p_buildings_delete", sqlConnection))
129             {
130                 cmd.CommandType = CommandType.StoredProcedure;
131                 cmd.Parameters.AddWithValue("@bui_id", id);
132                 try
133                 {
134                     sqlConnection.Open();
135                     cmd.ExecuteNonQuery();
136                 }
137                 finally
138                 {
139                     sqlConnection.Close();
140                 }
141             }
142         }
143
144         public IEnumerable<TFlats> TFlatsSelectAll()
145         {
146             List<TFlats> items = new List<TFlats>();
147             using (SqlCommand cmd = new SqlCommand("p_flats_select_all", sqlConnection))
148             {
149                 cmd.CommandType = CommandType.StoredProcedure;
150                 try
151                 {
152                     sqlConnection.Open();
153                     SqlDataReader rdr = cmd.ExecuteReader();
154                     while (rdr.Read())
155                     {
156                         TFlats item = new TFlats();
157                         item.FlaId = rdr.GetInt32("fla_id");
158                         item.FlaBuiId = rdr.GetInt32("fla_bui_id");
```

```
159                         item.FlaNumber = rdr["fla_number"].ToString();
160                         item.FlaOwner = rdr["fla_owner"].ToString();
161                         item.FlaAreaInM2 = ExtractDoubleValue(rdr, "fla_area_in_m2");
162                         item.FlaFloor = rdr.GetInt32("fla_floor");
163                         item.FlaFlatTypeDic = rdr["fla_flat_type_dic"].ToString();
164                         items.Add(item);
165                     }
166                 }
167                 finally
168                 {
169                     sqlConnection.Close();
170                 }
171             }
172             return items;
173         }
174 
175         public TFlats TFlatsSelectSingle(int id)
176         {
177             TFlats  item = new TFlats();
178             using (SqlCommand cmd = new SqlCommand("p_flats_select_single", sqlConnection))
179             {
180                 cmd.CommandType = CommandType.StoredProcedure;
181                 cmd.Parameters.AddWithValue("@fla_id", id);
182                 try
183                 {
184                     sqlConnection.Open();
185                     SqlDataReader rdr = cmd.ExecuteReader();
186                     if (rdr.Read())
187                     {
188                         item.FlaId = rdr.GetInt32("fla_id");
189                         item.FlaBuiId = rdr.GetInt32("fla_bui_id");
190                         item.FlaNumber = rdr["fla_number"].ToString();
191                         item.FlaOwner = rdr["fla_owner"].ToString();
192                         item.FlaAreaInM2 = ExtractDoubleValue(rdr, "fla_area_in_m2");
193                         item.FlaFloor = rdr.GetInt32("fla_floor");
194                         item.FlaFlatTypeDic = rdr["fla_flat_type_dic"].ToString();
195 
196                     }
197                 }
198                 finally
199                 {
200                     sqlConnection.Close();
201                 }
202             }
203             return item;
204         }
205 
206         public void TFlatsInsertUpdate(TFlats item)
207         {
208             using (SqlCommand cmd = new SqlCommand("p_flats_insert_update", sqlConnection))
209             {
210                 cmd.CommandType = CommandType.StoredProcedure;
211                 cmd.Parameters.AddWithValue("@fla_id", item.FlaId);
212                 cmd.Parameters.AddWithValue("@fla_bui_id", item.FlaBuiId);
```

```
213                 cmd.Parameters.AddWithValue("@fla_number", item.FlaNumber);
214                 cmd.Parameters.AddWithValue("@fla_owner", item.FlaOwner);
215                 cmd.Parameters.AddWithValue("@fla_area_in_m2", item.FlaAreaInM2);
216                 cmd.Parameters.AddWithValue("@fla_floor", item.FlaFloor);
217                 cmd.Parameters.AddWithValue("@fla_flat_type_dic", item.FlaFlatTypeDic);
218
219                 try
220                 {
221                     sqlConnection.Open();
222                     cmd.ExecuteNonQuery();
223                 }
224                 finally
225                 {
226                     sqlConnection.Close();
227                 }
228             }
229         }
230
231         public void TFlatsDelete(int id)
232         {
233             using (SqlCommand cmd = new SqlCommand("p_flats_delete", sqlConnection))
234             {
235                 cmd.CommandType = CommandType.StoredProcedure;
236                 cmd.Parameters.AddWithValue("@fla_id", id);
237                 try
238                 {
239                     sqlConnection.Open();
240                     cmd.ExecuteNonQuery();
241                 }
242                 finally
243                 {
244                     sqlConnection.Close();
245                 }
246             }
247         }
248
249         public IEnumerable<TRooms> TRoomsSelectAll()
250         {
251             List<TRooms> items = new List<TRooms>();
252             using (SqlCommand cmd = new SqlCommand("p_rooms_select_all", sqlConnection))
253             {
254                 cmd.CommandType = CommandType.StoredProcedure;
255                 try
256                 {
257                     sqlConnection.Open();
258                     SqlDataReader rdr = cmd.ExecuteReader();
259                     while (rdr.Read())
260                     {
261                         TRooms item = new TRooms();
262                         item.RooId = rdr.GetInt32("roo_id");
263                         item.RooFlaId = rdr.GetInt32("roo_fla_id");
264                         item.RooName = rdr["roo_name"].ToString();
265                         item.RooAreaInM2 = ExtractDoubleValue(rdr, "roo_area_in_m2");
```

```
266                         item.RooWindowsDirectionDic = rdr["roo_windows_direction_dic"].
    ToString();
267                         item.RooRoomTypeDic = rdr["roo_room_type_dic"].ToString();
268                         items.Add(item);
269                     }
270                 }
271                 finally
272                 {
273                     sqlConnection.Close();
274                 }
275             }
276             return items;
277         }
278 
279         public TRooms TRoomsSelectSingle(int id)
280         {
281             TRooms item = new TRooms();
282             using (SqlCommand cmd = new SqlCommand("p_rooms_select_single", sqlConnection))
283             {
284                 cmd.CommandType = CommandType.StoredProcedure;
285                 cmd.Parameters.AddWithValue("@roo_id", id);
286                 sqlConnection.Open();
287                 try
288                 {
289                     SqlDataReader rdr = cmd.ExecuteReader();
290                     if (rdr.Read())
291                     {
292                         item.RooId = rdr.GetInt32("roo_id");
293                         item.RooFlaId = rdr.GetInt32("roo_fla_id");
294                         item.RooName = rdr["roo_name"].ToString();
295                         item.RooAreaInM2 = ExtractDoubleValue(rdr, "roo_area_in_m2");
296                         item.RooWindowsDirectionDic = rdr["roo_windows_direction_dic"].
    ToString();
297                         item.RooRoomTypeDic = rdr["roo_room_type_dic"].ToString();
298 
299                     }
300                 }
301                 finally
302                 {
303                     sqlConnection.Close();
304                 }
305             }
306             return item;
307         }
308         public void TRoomsInsertUpdate(TRooms item)
309         {
310             using (SqlCommand cmd = new SqlCommand("p_rooms_insert_update", sqlConnection))
311             {
312                 cmd.CommandType = CommandType.StoredProcedure;
313                 cmd.Parameters.AddWithValue("@roo_id", item.RooId);
314                 cmd.Parameters.AddWithValue("@roo_fla_id", item.RooFlaId);
315                 cmd.Parameters.AddWithValue("@roo_name", item.RooName);
316                 cmd.Parameters.AddWithValue("@roo_area_in_m2", item.RooAreaInM2);
```

```
317                 cmd.Parameters.AddWithValue("@roo_windows_direction_dic", item.
    RooWindowsDirectionDic);
318                 cmd.Parameters.AddWithValue("@roo_room_type_dic", item.RooRoomTypeDic);
319                 try
320                 {
321                     sqlConnection.Open();
322                     cmd.ExecuteNonQuery();
323                 }
324                 finally
325                 {
326                     sqlConnection.Close();
327                 }
328             }
329         }
330         public void TRoomsDelete(int id)
331         {
332             using (SqlCommand cmd = new SqlCommand("p_rooms_delete", sqlConnection))
333             {
334                 cmd.CommandType = CommandType.StoredProcedure;
335                 cmd.Parameters.AddWithValue("@roo_id", id);
336                 try
337                 {
338                     sqlConnection.Open();
339                     cmd.ExecuteNonQuery();
340                 }
341                 finally
342                 {
343                     sqlConnection.Close();
344                 }
345             }
346         }
347
348         public IEnumerable<TSensors> TSensorsSelectAll()
349         {
350             List<TSensors> items = new List<TSensors>();
351             using (SqlCommand cmd = new SqlCommand("p_sensors_select_all", sqlConnection))
352             {
353                 cmd.CommandType = CommandType.StoredProcedure;
354                 try
355                 {
356                     sqlConnection.Open();
357                     SqlDataReader rdr = cmd.ExecuteReader();
358                     while (rdr.Read())
359                     {
360                         TSensors item = new TSensors();
361                         item.SenId = rdr.GetInt32("sen_id");
362                         item.SenTypeDic = rdr["sen_type_dic"].ToString();
363                         item.SenName = rdr["sen_name"].ToString();
364                         items.Add(item);
365                     }
366                 }
367                 finally
368                 {
369                     sqlConnection.Close();
```

```
370                 }
371             }
372             return items;
373         }
374         public TSensors TSensorsSelectSingle(int id)
375         {
376             TSensors item = new TSensors();
377             using (SqlCommand cmd = new SqlCommand("p_sensors_select_single", sqlConnection))
378             {
379                 cmd.CommandType = CommandType.StoredProcedure;
380                 cmd.Parameters.AddWithValue("@sen_id", id);
381                 try
382                 {
383                     sqlConnection.Open();
384                     SqlDataReader rdr = cmd.ExecuteReader();
385                     if (rdr.Read())
386                     {
387                         item.SenId = rdr.GetInt32("sen_id");
388                         item.SenTypeDic = rdr["sen_type_dic"].ToString();
389                         item.SenName = rdr["sen_name"].ToString();
390                     }
391                 }
392                 finally
393                 {
394                     sqlConnection.Close();
395                 }
396             }
397             return item;
398         }
399         public void TSensorsInsertUpdate(TSensors item)
400         {
401             using (SqlCommand cmd = new SqlCommand("p_sensors_insert_update", sqlConnection))
402             {
403                 cmd.CommandType = CommandType.StoredProcedure;
404                 cmd.Parameters.AddWithValue("@sen_id", item.SenId);
405                 cmd.Parameters.AddWithValue("@sen_type_dic", item.SenTypeDic);
406                 cmd.Parameters.AddWithValue("@sen_name", item.SenName);
407                 try
408                 {
409                     sqlConnection.Open();
410                     cmd.ExecuteNonQuery();
411                 }
412                 finally
413                 {
414                     sqlConnection.Close();
415                 }
416             }
417         }
418         public void TSensorsDelete(int id)
419         {
420             using (SqlCommand cmd = new SqlCommand("p_sensors_delete", sqlConnection))
421             {
422                 cmd.CommandType = CommandType.StoredProcedure;
423                 cmd.Parameters.AddWithValue("@sen_id", id);
```

```
424                 try
425                 {
426                     sqlConnection.Open();
427                     cmd.ExecuteNonQuery();
428                 }
429                 finally
430                 {
431                     sqlConnection.Close();
432                 }
433             }
434         }
435 
436         public IEnumerable<TInstalledSensors> TInstalledSensorsSelectAll()
437         {
438             List<TInstalledSensors> items = new List<TInstalledSensors>();
439             using (SqlCommand cmd = new SqlCommand("p_installed_sensors_select_all",
    sqlConnection))
440             {
441                 cmd.CommandType = CommandType.StoredProcedure;
442                 try
443                 {
444                     sqlConnection.Open();
445                     SqlDataReader rdr = cmd.ExecuteReader();
446                     while (rdr.Read())
447                     {
448                         TInstalledSensors item = new TInstalledSensors();
449                         item.InsId = rdr.GetInt32("ins_id");
450                         item.InsSenId = ExtractIntValue(rdr, "ins_sen_id");
451                         item.InsRooId = ExtractIntValue(rdr, "ins_roo_id");
452                         item.InsAuditCd = rdr.GetDateTime("ins_audit_cd");
453                         item.InsName = rdr["ins_name"].ToString();
454                         item.InsSerialNumber = rdr["ins_serial_number"].ToString();
455                         item.InsAccessKey = rdr["ins_access_key"].ToString();
456                         items.Add(item);
457                     }
458                 }
459                 finally
460                 {
461                     sqlConnection.Close();
462                 }
463             }
464             return items;
465         }
466         public TInstalledSensors TInstalledSensorsSelectSingle(int id)
467         {
468             TInstalledSensors item = new TInstalledSensors();
469             using (SqlCommand cmd = new SqlCommand("p_installed_sensors_select_single",
    sqlConnection))
470             {
471                 cmd.CommandType = CommandType.StoredProcedure;
472                 cmd.Parameters.AddWithValue("@ins_id", id);
473                 try
474                 {
475                     sqlConnection.Open();
```

```
476                     SqlDataReader rdr = cmd.ExecuteReader();
477                     if (rdr.Read())
478                     {
479                         item.InsId = rdr.GetInt32("ins_id");
480                         item.InsSenId = ExtractIntValue(rdr, "ins_sen_id");
481                         item.InsRooId = ExtractIntValue(rdr, "ins_roo_id");
482                         item.InsAuditCd = rdr.GetDateTime("ins_audit_cd");
483                         item.InsName = rdr["ins_name"].ToString();
484                         item.InsSerialNumber = rdr["ins_serial_number"].ToString();
485                         item.InsAccessKey = rdr["ins_access_key"].ToString();
486                     }
487                 }
488                 finally
489                 {
490                     sqlConnection.Close();
491                 }
492             }
493             return item;
494         }
495         public void TInstalledSensorsInsertUpdate(TInstalledSensors item)
496         {
497             using (SqlCommand cmd = new SqlCommand("p_installed_sensors_insert_update",
    sqlConnection))
498             {
499                 cmd.CommandType = CommandType.StoredProcedure;
500                 cmd.Parameters.AddWithValue("@ins_id", item.InsId);
501                 cmd.Parameters.AddWithValue("@ins_sen_id", item.InsSenId);
502                 cmd.Parameters.AddWithValue("@ins_roo_id", item.InsRooId);
503                 cmd.Parameters.AddWithValue("@ins_name", item.InsName);
504                 cmd.Parameters.AddWithValue("@ins_serial_number", item.InsSerialNumber);
505                 cmd.Parameters.AddWithValue("@ins_access_key", item.InsAccessKey);
506 
507                 try
508                 {
509                     sqlConnection.Open();
510                     cmd.ExecuteNonQuery();
511                 }
512                 finally
513                 {
514                     sqlConnection.Close();
515                 }
516             }
517         }
518         public void TInstalledSensorsDelete(int id)
519         {
520             using (SqlCommand cmd = new SqlCommand("p_installed_sensors_delete", sqlConnection
    ))
521             {
522                 cmd.CommandType = CommandType.StoredProcedure;
523                 cmd.Parameters.AddWithValue("@ins_id", id);
524                 try
525                 {
526                     sqlConnection.Open();
527                     cmd.ExecuteNonQuery();
```

```
528                 }
529                 finally
530                 {
531                     sqlConnection.Close();
532                 }
533             }
534         }
535         public IEnumerable<TReadouts> TReadoutsSelectAll()
536         {
537             List<TReadouts> items = new List<TReadouts>();
538             using (SqlCommand cmd = new SqlCommand("p_readouts_select_all", sqlConnection))
539             {
540                 cmd.CommandType = CommandType.StoredProcedure;
541                 try
542                 {
543                     sqlConnection.Open();
544                     SqlDataReader rdr = cmd.ExecuteReader();
545                     while (rdr.Read())
546                     {
547                         TReadouts item = new TReadouts();
548                         item.ReaId = rdr.GetInt32("rea_id");
549                         item.ReaInsId = rdr.GetInt32("rea_ins_id");
550                         item.ReaAuditCd = rdr.GetDateTime("rea_audit_cd");
551                         item.ReaData = ExtractDoubleValue(rdr, "rea_data");
552                         item.ReaSequence = ExtractByteValue(rdr, "rea_sequence");
553                         item.ReaTemperature = ExtractFloatValue(rdr, "rea_temperature");
554                         item.ReaHumidity = ExtractFloatValue(rdr, "rea_humidity");
555                         item.ReaVoltage = ExtractFloatValue(rdr, "rea_voltage");
556                         item.ReaTimestamp = ExtractDateTimeValue(rdr, "rea_timestamp");
557                         item.ReaRSSI = ExtractShortValue(rdr, "rea_rssi");
558                         items.Add(item);
559                     }
560                 }
561                 finally
562                 {
563                     sqlConnection.Close();
564                 }
565             }
566             return items;
567         }
568         public TReadouts TReadoutsSelectSingle(int id)
569         {
570             TReadouts item = new TReadouts();
571             using (SqlCommand cmd = new SqlCommand("p_readouts_select_single", sqlConnection))
572             {
573                 cmd.CommandType = CommandType.StoredProcedure;
574                 cmd.Parameters.AddWithValue("@rea_id", id);
575                 try
576                 {
577                     sqlConnection.Open();
578                     SqlDataReader rdr = cmd.ExecuteReader();
579                     if (rdr.Read())
580                     {
581                         item.ReaId = rdr.GetInt32("rea_id");
```

```
582                         item.ReaInsId = rdr.GetInt32("rea_ins_id");
583                         item.ReaAuditCd = rdr.GetDateTime("rea_audit_cd");
584                         item.ReaData = ExtractDoubleValue(rdr, "rea_data");
585                         item.ReaSequence = ExtractByteValue(rdr, "rea_sequence");
586                         item.ReaTemperature = ExtractFloatValue(rdr, "rea_temperature");
587                         item.ReaHumidity = ExtractFloatValue(rdr, "rea_humidity");
588                         item.ReaVoltage = ExtractFloatValue(rdr, "rea_voltage");
589                         item.ReaTimestamp = ExtractDateTimeValue(rdr, "rea_timestamp");
590                         item.ReaRSSI = ExtractShortValue(rdr, "rea_rssi");
591                     }
592                 }
593                 finally
594                 {
595                     sqlConnection.Close();
596                 }
597             }
598             return item;
599         }
600 
601         public void TReadoutsInsertUpdate(TReadouts item)
602         {
603             using (SqlCommand cmd = new SqlCommand("p_readouts_insert_update", sqlConnection))
604             {
605                 cmd.CommandType = CommandType.StoredProcedure;
606                 cmd.Parameters.AddWithValue("@rea_id", item.ReaId);
607                 cmd.Parameters.AddWithValue("@rea_ins_id", item.ReaInsId);
608                 cmd.Parameters.AddWithValue("@rea_data", item.ReaData);
609                 cmd.Parameters.AddWithValue("@rea_sequence", item.ReaSequence);
610                 cmd.Parameters.AddWithValue("@rea_temperature", item.ReaTemperature);
611                 cmd.Parameters.AddWithValue("@rea_humidity", item.ReaHumidity);
612                 cmd.Parameters.AddWithValue("@rea_voltage", item.ReaVoltage);
613                 cmd.Parameters.AddWithValue("@rea_timestamp", item.ReaTimestamp);
614                 cmd.Parameters.AddWithValue("@rea_rssi", item.ReaRSSI);
615                 try
616                 {
617                     sqlConnection.Open();
618                     cmd.ExecuteNonQuery();
619                 }
620                 finally
621                 {
622                     sqlConnection.Close();
623                 }
624             }
625         }
626 
627         public void TReadoutsDelete(int id)
628         {
629             using (SqlCommand cmd = new SqlCommand("p_readouts_delete", sqlConnection))
630             {
631                 cmd.CommandType = CommandType.StoredProcedure;
632                 cmd.Parameters.AddWithValue("@rea_id", id);
633                 try
634                 {
635                     sqlConnection.Open();
```

```
636                     cmd.ExecuteNonQuery();
637                 }
638                 finally
639                 {
640                     sqlConnection.Close();
641                 }
642             }
643         }
644 
645         public IEnumerable<TDictionaries> TDictionariesSelectAll()
646         {
647             List<TDictionaries> items = new List<TDictionaries>();
648             using (SqlCommand cmd = new SqlCommand("p_dictionaries_select_all", sqlConnection)
    )
649             {
650                 cmd.CommandType = CommandType.StoredProcedure;
651                 try
652                 {
653                     sqlConnection.Open();
654                     SqlDataReader rdr = cmd.ExecuteReader();
655                     while (rdr.Read())
656                     {
657                         TDictionaries item = new TDictionaries();
658                         item.DicId = rdr.GetInt32("dic_id");
659                         item.DicGroup = rdr["dic_group"].ToString();
660                         item.DicKey = rdr["dic_key"].ToString();
661                         item.DicValue = rdr["dic_value"].ToString();
662                         items.Add(item);
663                     }
664                 }
665                 finally
666                 {
667                     sqlConnection.Close();
668                 }
669             }
670             return items;
671         }
672 
673         public TDictionaries TDictionariesSelectSingle(int id)
674         {
675             TDictionaries item = new TDictionaries();
676             using (SqlCommand cmd = new SqlCommand("p_dictionaries_select_single",
    sqlConnection))
677             {
678                 cmd.CommandType = CommandType.StoredProcedure;
679                 cmd.Parameters.AddWithValue("@dic_id", id);
680                 sqlConnection.Open();
681                 try
682                 {
683                     SqlDataReader rdr = cmd.ExecuteReader();
684                     if (rdr.Read())
685                     {
686                         item.DicId = rdr.GetInt32("dic_id");
687                         item.DicGroup = rdr["dic_group"].ToString();
```

```
688                         item.DicKey = rdr["dic_key"].ToString();
689                         item.DicValue = rdr["dic_value"].ToString();
690                     }
691                 }
692                 finally
693                 {
694                     sqlConnection.Close();
695                 }
696             }
697             return item;
698         }
699 
700         public void TDictionariesInsertUpdate(TDictionaries item)
701         {
702             using (SqlCommand cmd = new SqlCommand("p_dictionaries_insert_update",
    sqlConnection))
703             {
704                 cmd.CommandType = CommandType.StoredProcedure;
705                 cmd.Parameters.AddWithValue("@dic_id", item.DicId);
706                 cmd.Parameters.AddWithValue("@dic_group", item.DicGroup);
707                 cmd.Parameters.AddWithValue("@dic_key", item.DicKey);
708                 cmd.Parameters.AddWithValue("@dic_value", item.DicValue);
709 
710                 try
711                 {
712                     sqlConnection.Open();
713                     cmd.ExecuteNonQuery();
714                 }
715                 finally
716                 {
717                     sqlConnection.Close();
718                 }
719             }
720         }
721 
722         public void TDictionariesDelete(int id)
723         {
724             using (SqlCommand cmd = new SqlCommand("p_dictionaries_delete", sqlConnection))
725             {
726                 cmd.CommandType = CommandType.StoredProcedure;
727                 cmd.Parameters.AddWithValue("@dic_id", id);
728                 try
729                 {
730                     sqlConnection.Open();
731                     cmd.ExecuteNonQuery();
732                 }
733                 finally
734                 {
735                     sqlConnection.Close();
736                 }
737             }
738         }
739 
```

```
740         public IEnumerable<VReadouts> VReadoutsSelect(DateTime? rea_date_from, DateTime?
    rea_date_to)
741         {
742             List<VReadouts> items = new List<VReadouts>();
743             using (SqlCommand cmd = new SqlCommand("p_sensor_data_select", sqlConnection))
744             {
745                 if (rea_date_from.HasValue)
746                 {
747                     cmd.Parameters.AddWithValue("@rea_date_from", rea_date_from);
748                 }
749                 if (rea_date_to.HasValue)
750                 {
751                     cmd.Parameters.AddWithValue("@rea_date_to", rea_date_to);
752                 }
753                 cmd.CommandType = CommandType.StoredProcedure;
754                 try
755                 {
756                     sqlConnection.Open();
757                     SqlDataReader rdr = cmd.ExecuteReader();
758                     while (rdr.Read())
759                     {
760                         VReadouts item = new VReadouts();
761                         item.BuiId = rdr.GetInt32("bui_id");
762                         item.FlaId = rdr.GetInt32("fla_id");
763                         item.RooId = rdr.GetInt32("roo_id");
764                         item.InsId = rdr.GetInt32("ins_id");
765                         item.ReaAuditCd = rdr.GetDateTime("rea_audit_cd");
766                         item.ReaTemperature = ExtractFloatValue(rdr, "rea_temperature");
767                         item.SenTypeDic = rdr.GetString("sen_type_dic");
768                         items.Add(item);
769                     }
770                 }
771                 finally
772                 {
773                     sqlConnection.Close();
774                 }
775             }
776             return items;
777         }
778
779         public IEnumerable<TParameters> TParametersSelectAll()
780         {
781             List<TParameters> items = new List<TParameters>();
782             using (SqlCommand cmd = new SqlCommand("p_parameters_select_all", sqlConnection))
783             {
784                 cmd.CommandType = CommandType.StoredProcedure;
785                 try
786                 {
787                     sqlConnection.Open();
788                     SqlDataReader rdr = cmd.ExecuteReader();
789                     while (rdr.Read())
790                     {
791                         TParameters item = new TParameters();
792                         item.ParId = rdr.GetInt32("par_id");
```

```
793                         item.ParName = rdr["par_name"].ToString();
794                         item.ParValue = rdr["par_value"].ToString();
795                         item.ParDescription = rdr["par_description"].ToString();
796                         items.Add(item);
797                     }
798                 }
799                 finally
800                 {
801                     sqlConnection.Close();
802                 }
803             }
804             return items;
805         }
806         public TParameters TParametersSelectSingle(int id)
807         {
808             TParameters item = new TParameters();
809             using (SqlCommand cmd = new SqlCommand("p_parameters_select_single", sqlConnection
    ))
810             {
811                 cmd.CommandType = CommandType.StoredProcedure;
812                 cmd.Parameters.AddWithValue("@par_id", id);
813                 try
814                 {
815                     sqlConnection.Open();
816                     SqlDataReader rdr = cmd.ExecuteReader();
817                     if (rdr.Read())
818                     {
819                         item.ParId = rdr.GetInt32("par_id");
820                         item.ParName = rdr["par_name"].ToString();
821                         item.ParValue = rdr["par_value"].ToString();
822                         item.ParDescription = rdr["par_description"].ToString();
823                     }
824                 }
825                 finally
826                 {
827                     sqlConnection.Close();
828                 }
829             }
830             return item;
831         }
832         public void TParametersInsertUpdate(TParameters item)
833         {
834             using (SqlCommand cmd = new SqlCommand("p_parameters_insert_update", sqlConnection
    ))
835             {
836                 cmd.CommandType = CommandType.StoredProcedure;
837                 cmd.Parameters.AddWithValue("@par_id", item.ParId);
838                 cmd.Parameters.AddWithValue("@par_name", item.ParName);
839                 cmd.Parameters.AddWithValue("@par_value", item.ParValue);
840                 cmd.Parameters.AddWithValue("@par_description", item.ParDescription);
841 
842                 try
843                 {
844                     sqlConnection.Open();
```

```
845                     cmd.ExecuteNonQuery();
846                 }
847                 finally
848                 {
849                     sqlConnection.Close();
850                 }
851             }
852         }
853         public void TParametersDelete(int id)
854         {
855             using (SqlCommand cmd = new SqlCommand("p_parameters_delete", sqlConnection))
856             {
857                 cmd.CommandType = CommandType.StoredProcedure;
858                 cmd.Parameters.AddWithValue("@par_id", id);
859                 try
860                 {
861                     sqlConnection.Open();
862                     cmd.ExecuteNonQuery();
863                 }
864                 finally
865                 {
866                     sqlConnection.Close();
867                 }
868             }
869         }
870 
871         public static decimal? ExtractDecimalValue(SqlDataReader rdr, String column)
872         {
873             if (!rdr.IsDBNull(column))
874                 return rdr.GetDecimal(column);
875             return null;
876         }
877 
878         public static double? ExtractDoubleValue(SqlDataReader rdr, String column)
879         {
880             if (!rdr.IsDBNull(column))
881                 return rdr.GetDouble(column);
882             return null;
883         }
884         public static float? ExtractFloatValue(SqlDataReader rdr, String column)
885         {
886             if (!rdr.IsDBNull(column))
887                 return rdr.GetFloat(column);
888             return null;
889         }
890 
891         public static int? ExtractIntValue(SqlDataReader rdr, String column)
892         {
893             if (!rdr.IsDBNull(column))
894                 return rdr.GetInt32(column);
895             return null;
896         }
897 
898         public static int? ExtractByteValue(SqlDataReader rdr, String column)
```

```
899         {
900             if (!rdr.IsDBNull(column))
901                 return rdr.GetByte(column);
902             return null;
903         }
904 
905         public static int? ExtractShortValue(SqlDataReader rdr, String column)
906         {
907             if (!rdr.IsDBNull(column))
908                 return rdr.GetInt16(column);
909             return null;
910         }
911 
912         public static bool? ExtractBooleanValue(SqlDataReader rdr, String column)
913         {
914             if (!rdr.IsDBNull(column))
915                 return rdr.GetBoolean(column);
916             return null;
917         }
918 
919         public static DateTime? ExtractDateTimeValue(SqlDataReader rdr, String column)
920         {
921             if (!rdr.IsDBNull(column))
922                 return rdr.GetDateTime(column);
923             return null;
924         }
925 
926         public void Dispose()
927         {
928 
929         }
930     }
931 }
```

B.3. Data cache service

Listing B.3. Data cache service class

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using DataCloud.Models;
6  using Microsoft.Extensions.Configuration;
7  using Microsoft.Extensions.DependencyInjection;
8  using Microsoft.Extensions.Hosting;
9  using System.Data;
10 using System.Data.SqlClient;
11 using System.IO;
12 
13 namespace DataCloud.Data
14 {
15     public class DBCacheService : IDisposable
```

```
16      {
17          private readonly IConfiguration configuration;
18          private readonly DataAccessService das;
19
20          public DBCacheService(IConfiguration configuration, DataAccessService
    dataAccessService)
21          {
22              this.configuration = configuration;
23              this.das = dataAccessService;
24              Invalidate();
25          }
26
27          public Dictionary<string, string> Parameters {
28              get
29              {
30                  return parameters;
31              }
32          }
33
34          public Dictionary<int, string> SensorIdToTypeDic
35          {
36              get
37              {
38                  return sensorIdToTypeDic;
39              }
40          }
41          Dictionary<int, string> sensorIdToTypeDic = new Dictionary<int, string>();
42          Dictionary<string, string> parameters = new Dictionary<string, string>();
43          Dictionary<string, Dictionary<string, string>> dictionaries =
44              new Dictionary<string, Dictionary<string, string>>();
45
46          public const string KEY_NOT_SELECTED = "?";
47          public static class DictionaryTypes
48          {
49              public const string BuildingType = "BUILDING_TYPE";
50              public const string FlatType = "FLAT_TYPE";
51              public const string RoomType = "ROOM_TYPE";
52              public const string WindowsDirection = "WINDOWS_DIRECTION";
53          }
54
55          public static class ParameterNames
56          {
57              public const string READOUTS_SERVER_IP = "READOUTS_SERVER_IP";
58              public const string SAVE_READOUTS_TO_DB = "SAVE_READOUTS_TO_DB";
59              public const string READOUTS_QUEUE_LENGTH = "READOUTS_QUEUE_LENGTH";
60          }
61
62          public Dictionary<string, string> Dictionary(string dictionaryName)
63          {
64              if (!dictionaries.ContainsKey(dictionaryName))
65              {
66                  throw new Exception($"Unknown dictionary: {dictionaryName}");
67              }
68              return dictionaries[dictionaryName];
```

```
69          }
70
71          private void BuildDictionaries()
72          {
73              dictionaries.Clear();
74              foreach (TDictionaries dic in das.TDictionariesSelectAll())
75              {
76                  if (string.IsNullOrWhiteSpace(dic.DicGroup) ||
77                      string.IsNullOrWhiteSpace(dic.DicKey) ||
78                      string.IsNullOrWhiteSpace(dic.DicValue))
79                  {
80                      continue;
81                  }
82                  if (!dictionaries.ContainsKey(dic.DicGroup))
83                  {
84                      dictionaries.Add(dic.DicGroup, new Dictionary<string, string>());
85                  }
86                  (dictionaries[dic.DicGroup])[dic.DicKey] = dic.DicValue;
87              }
88          }
89
90          private void BuildParameters()
91          {
92              parameters.Clear();
93              foreach (TParameters param in das.TParametersSelectAll())
94              {
95                  if (string.IsNullOrWhiteSpace(param.ParName))
96                  {
97                      continue;
98                  }
99                  parameters[param.ParName] = param.ParValue;
100             }
101         }
102
103         private void BuildSensorIdToTypeDic()
104         {
105             sensorIdToTypeDic.Clear();
106             foreach (TSensors item in das.TSensorsSelectAll())
107             {
108                 sensorIdToTypeDic[item.SenId] = item.SenTypeDic;
109             }
110         }
111         public void Invalidate()
112         {
113             BuildDictionaries();
114             BuildParameters();
115             BuildSensorIdToTypeDic();
116         }
117
118         public void Dispose()
119         {
120
121         }
122     }
```

```
}
```

B.4. Transport objects

Listing B.4. Data transfer object for sensors

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace DataCloud.Models
{
    public partial class TSensors
    {
        public int SenId { get; set; }
        [Display(Name = "Type dic")]
        public string SenTypeDic { get; set; }
        [Display(Name = "Name")]
        public string SenName { get; set; }
    }
}
```

Listing B.5. Data transfer object for installed sensors

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace DataCloud.Models
{
    public partial class TInstalledSensors
    {
        public int InsId { get; set; }
        [Display(Name = "Sensors id")]
        public int? InsSenId { get; set; }
        [Display(Name = "Rooms id")]
        public int? InsRooId { get; set; }
        [Display(Name = "Audit cd")]
        public DateTime InsAuditCd { get; set; }
        [Display(Name = "Name")]
        public string InsName { get; set; }
        [Display(Name = "Serial number")]
        public string InsSerialNumber { get; set; }
        [Display(Name = "Access key")]
        public string InsAccessKey { get; set; }
    }
}
```

Listing B.6. Data transfer object for readouts

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
```

```
4
5  namespace DataCloud.Models
6  {
7      public partial class TReadouts
8      {
9          public int ReaId { get; set; }
10         [Display(Name = "Installed sensor id")]
11         public int ReaInsId { get; set; }
12         [Display(Name = "Audit cd")]
13         public DateTime ReaAuditCd { get; set; }
14         [Display(Name = "Data")]
15         public double? ReaData { get; set; }
16         public int? ReaSequence { get; set; }
17         public float? ReaTemperature { get; set; }
18         public float? ReaHumidity { get; set; }
19         public float? ReaVoltage { get; set; }
20         public int? ReaRSSI { get; set; }
21         public DateTime? ReaTimestamp { get; set; }
22     }
23 }
```

Listing B.7. Data transfer object for sensor view

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.DataAnnotations;
4
5  namespace DataCloud.Models
6  {
7      public partial class VSensors
8      {
9          public int SenId { get; set; }
10         public string SenTypeDic { get; set; }
11         public string SenName { get; set; }
12         public int ReaId { get; set; }
13         public DateTime ReaAuditCd { get; set; }
14         public double? ReaData { get; set; }
15     }
16 }
```

Listing B.8. Data transfer object for sensor data

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace DataCloud.Models
7  {
8      public partial class VSensorData
9      {
10         public string Name { get; set; }
11         public DateTime Time { get; set; }
12         public double Value { get; set; }
```

```
13          public int? Sequence { get; set; }
14          public float? Temperature { get; set; }
15          public float? Humidity { get; set; }
16          public float? Voltage { get; set; }
17          public int? RSSI { get; set; }
18          public DateTime? Timestamp { get; set; }
19      }
20  }
```

B.5. Controller

Listing B.9. Main controller for sensors

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Http;
6  using Microsoft.AspNetCore.Mvc;
7  using Microsoft.Extensions.Configuration;
8  using Microsoft.Extensions.Options;
9  using DataCloud.Data;
10 using DataCloud.Models;
11 using DataCloud.Hubs;
12 using Microsoft.AspNetCore.SignalR;
13
14 namespace DataCloud.Controllers
15 {
16     public static class ErrorCodes
17     {
18         public const int OK = 0;
19         public const int DB = 1;
20         public const int INVALID_VERSION = 2;
21     }
22
23     public class DataCloudBaseRequest
24     {
25         public string Key { get; set; }
26     }
27
28     public class DataCloudRequest : DataCloudBaseRequest
29     {
30         public string Version { get; set; }
31         public int No { get; set; }
32         public DateTime Date { get; set; }
33     }
34
35     public class DataCloudUpdateRequest : DataCloudBaseRequest
36     {
37         public string SerialNumber { get; set; }
38         public double Value { get; set; }
39     }
40
```

```
41
42      public class DataCloudBaseResponse
43      {
44          public int ErrorCode { get; set; }
45          public string ErrorDesc { get; set; }
46      }
47
48      public class DataCloudResponse : DataCloudBaseResponse
49      {
50          public string Version { get; set; }
51          public int No { get; set; }
52          public SensorData[] SensorData { get; set; }
53      }
54
55      public class DataCloudReadoutRequest
56      {
57          public string Version { get; set; }
58          public string Id { get; set; }
59          public int? Sequence { get; set; }
60          public float? Temperature { get; set; }
61          public float? Humidity { get; set; }
62          public float? Voltage { get; set; }
63          public int? RSSI { get; set; }
64          public DateTime? Timestamp { get; set; }
65      }
66
67
68      public class SensorData
69      {
70          public int Building { get; set; }
71          public int Flat { get; set; }
72          public int Room { get; set; }
73          public int Sensor { get; set; }
74          public string Type { get; set; }
75          public double? Value { get; set; }
76          public DateTime Date { get; set; }
77      }
78
79      [Route("api/[controller]")]
80      [ApiController]
81      public class SensorsController : Controller
82      {
83          private readonly IConfiguration configuration;
84          private readonly DataAccessService das;
85          private readonly SensorStorageService sensorStorageService;
86          private readonly DBCacheService dbCacheService;
87          private readonly IHubContext<SignalRHub> signalRHubContext;
88
89          public SensorsController(IConfiguration configuration, IHubContext<SignalRHub>
    signalRHubContextToSet,
90              SensorStorageService sensorStorageService, DBCacheService dbCacheServiceToSet,
    DataAccessService dataAccessService)
91          {
92              this.configuration = configuration;
```

```
93              this.das = dataAccessService;
94              this.signalRHubContext = signalRHubContextToSet;
95              this.sensorStorageService = sensorStorageService;
96              this.dbCacheService = dbCacheServiceToSet;
97          }
98
99          [HttpGet]
100         public DataCloudRequest Get()
101         {
102             DataCloudRequest request = new DataCloudRequest
103             {
104                 Version = "1.0",
105                 No = Math.Abs(new Random().Next()),
106                 Date = DateTime.Now.Date,
107                 Key = System.Guid.NewGuid().ToString(),
108             };
109             return request;
110         }
111
112         [HttpPost]
113         public DataCloudResponse Post([FromBody] DataCloudRequest request)
114         {
115             /*
116             DataCloudRequest request = new DataCloudRequest
117             {
118                 Version = "1.0",
119                 No = Math.Abs(new Random().Next()),
120             };
121             */
122             DateTime dateFrom = request.Date == null ? DateTime.Today.Date : request.Date;
123             DateTime dateTo = dateFrom.AddDays(1);
124             DataCloudResponse response = new DataCloudResponse
125             {
126                 Version = request.Version,
127                 No = request.No,
128                 ErrorCode = ErrorCodes.OK,
129                 ErrorDesc = "OK"
130             };
131
132             if (!"1.0".Equals(request.Version))
133             {
134                 response.ErrorCode = ErrorCodes.INVALID_VERSION;
135                 response.ErrorDesc = "The only supported version is 1.0";
136             }
137             else
138             {
139                 try
140                 {
141                     LinkedList<SensorData> sds = new LinkedList<SensorData>();
142                     foreach (VReadouts item in das.VReadoutsSelect(dateFrom, dateTo))
143                     {
144                         SensorData sd = new SensorData();
145                         sd.Building = item.BuiId;
146                         sd.Flat = item.FlaId;
```

```
147                         sd.Room = item.RooId;
148                         sd.Sensor = item.InsId;
149                         sd.Value = item.ReaTemperature;
150                         sd.Date = item.ReaAuditCd;
151                         sd.Type = item.SenTypeDic;
152                         sds.AddLast(sd);
153                     }
154                     response.SensorData = sds.ToArray();
155                 }
156                 catch (Exception e)
157                 {
158                     response.ErrorDesc = e.Message;
159                     response.ErrorCode = ErrorCodes.DB;
160                 }
161             }
162             return response;
163         }
164 
165         //http://host:port/api/sensors/readout
166         [HttpPost("[action]")]
167         public DataCloudBaseResponse Readout([FromBody] DataCloudReadoutRequest request)
168         {
169             DataCloudBaseResponse response = new DataCloudBaseResponse
170             {
171                 ErrorCode = ErrorCodes.OK,
172                 ErrorDesc = "OK"
173             };
174             if (!"1.0".Equals(request.Version))
175             {
176                 response.ErrorCode = ErrorCodes.INVALID_VERSION;
177                 response.ErrorDesc = "The only supported version is 1.0";
178             }
179             else
180             {
181                 try
182                 {
183                     string remoteIpAddress = HttpContext.Connection.RemoteIpAddress?.ToString
    ();
184                     string allowedIps = dbCacheService.Parameters[DBCacheService.
    ParameterNames.READOUTS_SERVER_IP];
185                     if (!"*".Equals(allowedIps) && !allowedIps.Contains(remoteIpAddress))
186                     {
187                         throw new Exception($"Requests from {remoteIpAddress} not allowed!");
188                     }
189                     if (!request.Temperature.HasValue)
190                     {
191                         throw new MissingFieldException("Sensor data is missing!");
192                     }
193                     DateTime timestamp;
194                     if (request.Timestamp.HasValue)
195                     {
196                         timestamp = request.Timestamp.Value;
197                     }
198                     else
```

```
199                     {
200                         timestamp = DateTime.Now;
201                     }
202                     string valueString;
203                     double value = request.Temperature.Value;
204                     if (request.Sequence.HasValue && request.RSSI.HasValue)
205                     {
206                         valueString = String.Format("{0:0.0} ({1} {2})",
207                             request.Temperature.Value,
208                             request.Sequence.Value,
209                             request.RSSI.Value);
210                     }
211                     else
212                     {
213                         valueString = String.Format("{0:0.00}", value);
214                     }
215                     TReadouts readout = new TReadouts()
216                     {
217                         ReaSequence = request.Sequence,
218                         ReaTemperature = request.Temperature,
219                         ReaHumidity = request.Humidity,
220                         ReaVoltage = request.Voltage,
221                         ReaRSSI = request.RSSI,
222                         ReaTimestamp = request.Timestamp
223                     };
224                     sensorStorageService.AddReadout(request.Id, readout);
225                     signalRHubContext.Clients.All.SendAsync("ReceiveMessage", request.Id,
    timestamp, valueString);
226                     signalRHubContext.Clients.All.SendAsync("ReceiveReadout", request.Id);
227                 }
228                 catch (Exception e)
229                 {
230                     response.ErrorDesc = e.Message;
231                     response.ErrorCode = ErrorCodes.DB;
232                 }
233             }
234             return response;
235         }
236     }
237 }
```

B.6. Views

Listing B.10. Razor web page

```
1 @page "/"
2
3 @using DataCloud.Hubs;
4 @using DataCloud.Data;
5 @using DataCloud.Models;
6 @using Microsoft.AspNetCore.SignalR.Client
7 @inject NavigationManager NavigationManager
8 @inject SensorStorageService SensorStorageService
```

```
9
10 @implements IDisposable
11
12 <h1>Sensor data</h1>
13 @*
14     <p>Go to <a href="/api/sensors">/api/sensors</a> to retrieve sensor data.</p>
15
16     <p>
17         To send a HTTP request you can use CURL.
18         Sample request:
19         <pre>
20         curl -i -X POST -H "Content-Type: application/json" -d "{\"version\":\"1.0\",\"no\":@(
       randomNo),\"key\":\"9b6b9432\",\"date\":\"2020-06-04T00:00:00+02:00\"}" @(
       NavigationManager.Uri)api/sensors
21         </pre>
22     </p>
23 *@
24
25 @if (items != null)
26 {
27     <table class="table">
28         <thead>
29             <tr>
30                 <th class="la">Sensor</th>
31                 <th class="ra"><a href="/plot/temperature">Temperature</a></th>
32                 <th class="ra"><a href="/plot/humidity">Humidity</a></th>
33                 <th class="ra"><a href="/plot/voltage">Voltage</a></th>
34                 <th class="ra">Sequence</th>
35                 <th class="ra"><a href="/plot/signal">Signal</a></th>
36                 <th class="ca">Date</th>
37                 <th class="ca">Time</th>
38             </tr>
39         </thead>
40         <tbody>
41             @foreach (VSensorData item in items.Values)
42             {
43             <tr>
44                 <td class="la"><a href="@($"/sensor/{@item.Name}")">@item.Name</a></td>
45                 <td class="ra">@(item.Temperature.HasValue ? item.Temperature.Value.ToString("
       0.0") : "")</td>
46                 <td class="ra">@(item.Humidity)</td>
47                 <td class="ra">@(item.Voltage.HasValue ? item.Voltage.Value.ToString("0.00") :
        "")</td>
48                 <td class="ra">@(item.Sequence)</td>
49                 <td class="ra">@(item.RSSI)</td>
50                 <td class="ca">@(item.Timestamp.HasValue ? string.Format("{0:yyyy-MM-dd}",
       item.Timestamp) : "")</td>
51                 <td class="ca">@(item.Timestamp.HasValue ? string.Format("{0:HH:mm:ss}", item.
       Timestamp) : "")</td>
52             </tr>
53             }
54         </tbody>
55     </table>
56 }
```

```
@if (!string.IsNullOrEmpty(errorMessage))
{
    <p>@errorMessage</p>
}

@code {
    private HubConnection hubConnection;
    private Dictionary<string, VSensorData> items;
    private int randomNo = Math.Abs(new Random().Next());
    private string errorMessage;

    protected override async Task OnInitializedAsync()
    {
        items = SensorStorageService.SensorData;
        hubConnection = new HubConnectionBuilder()
            .WithUrl(NavigationManager.ToAbsoluteUri(SignalRHub.HUB_NAME))
            .WithUrl(NavigationManager.ToAbsoluteUri(SignalRHub.HUB_NAME).ToString().Replace("http://", "https://"))
            .Build();
        hubConnection.On<string>("ReceiveReadout", (sensorId) =>
        {
            try
            {
                StateHasChanged();
            }
            catch
            {
                //Log
            }
        });
        try
        {
            await hubConnection.StartAsync();
        }
        catch (Exception e)
        {
            errorMessage = e.Message;
        }
    }

    public bool IsConnected =>
        hubConnection.State == HubConnectionState.Connected;

    public void Dispose()
    {
        if (hubConnection != null)
        {
            _ = hubConnection.DisposeAsync();
        }
    }
}
```

Listing B.11. Razor web page for single sensor data

```
@page "/sensor/{Id}"

@using DataCloud.Hubs;
@using DataCloud.Data;
@using DataCloud.Models;
@using Microsoft.AspNetCore.SignalR.Client
@inject NavigationManager NavigationManager
@inject SensorStorageService SensorStorageService

@implements IDisposable

@if (item != null)
{

    <div style="font-size: 70pt;">@(item.Temperature.HasValue ? String.Format("{0:0.0}", item.
    Temperature.Value) : "_")<span style="vertical-align: top; font-size: 35pt; position:
    relative; top: 12pt; ">&deg;C</span></div>
    <div>
        <span class="sensortag">
            <span class="oi oi-droplet" aria-hidden="true"></span> @item.Humidity<text>%</text
    >
        </span>
        <span class="sensortag">
            <span class="oi oi-flash" aria-hidden="true"></span>
            @item.Voltage<text>V</text>
        </span>
        <span class="sensortag">
            <span class="oi oi-pulse" aria-hidden="true"></span>
            @item.Sequence
        </span>
        <span class="sensortag">
            <span class="oi oi-signal" aria-hidden="true"></span>
            @item.RSSI<text>dBm</text>
        </span>
        @if (item.Timestamp.HasValue)
        {
            <span class="sensortag">
                <span class="oi oi-calendar" aria-hidden="true"></span>
                @string.Format("{0:yyyy-MM-dd}", item.Timestamp)
            </span>
            <span class="sensortag">
                <span class="oi oi-clock" aria-hidden="true"></span>
                @string.Format("{0:HH:mm}", item.Timestamp)
            </span>
        }
    </div>
}
@if (!string.IsNullOrEmpty(errorMessage))
{
    <p>@errorMessage</p>
}

@code {
    private HubConnection hubConnection;
```

```
52      private VSensorData item;
53      private string errorMessage;
54
55      [Parameter]
56      public String Id { get; set; }
57
58      protected override async Task OnInitializedAsync()
59      {
60          if (SensorStorageService.SensorData.ContainsKey(Id))
61          {
62              item = SensorStorageService.SensorData[Id];
63          }
64
65          hubConnection = new HubConnectionBuilder()
66              .WithUrl(NavigationManager.ToAbsoluteUri(SignalRHub.HUB_NAME))
67              .WithUrl(NavigationManager.ToAbsoluteUri(SignalRHub.HUB_NAME).ToString().Replace("
    http://", "https://"))
68              .Build();
69          hubConnection.On<string>("ReceiveReadout", (sensorId) =>
70          {
71              if (!string.IsNullOrEmpty(Id) && Id.Equals(sensorId))
72              {
73                  if (item == null && SensorStorageService.SensorData.ContainsKey(Id))
74                  {
75                      item = SensorStorageService.SensorData[Id];
76                  }
77                  StateHasChanged();
78              }
79          });
80          try
81          {
82              await hubConnection.StartAsync();
83          }
84          catch (Exception e)
85          {
86              errorMessage = e.Message;
87          }
88      }
89
90      public bool IsConnected =>
91          hubConnection.State == HubConnectionState.Connected;
92
93      public void Dispose()
94      {
95          if (hubConnection != null)
96          {
97              _ = hubConnection.DisposeAsync();
98          }
99      }
100 }
```

Listing B.12. Razor web page for sensor readouts

```
1 @page "/readouts/"
```

```
2
3  @using Microsoft.AspNetCore.Identity;
4  @using DataCloud.Models;
5
6  @inject DataCloud.Data.DataAccessService DataAccessService
7
8  <h1>Readouts</h1>
9
10 <p><a class="btn btn-sm btn-outline-primary" href="/readouts/edit/0" role="button"><span class
       ="oi oi-plus" aria-hidden="true"></span> Add new</a></p>
11
12 <table class="table">
13     <thead>
14         <tr>
15             <th>Installed sensor id</th>
16             <th>Date</th>
17             <th>Data</th>
18             <th>Sequence</th>
19             <th>Temperature</th>
20             <th>Humidity</th>
21             <th>Voltage</th>
22             <th>RSSI</th>
23             <th>Action</th>
24         </tr>
25     </thead>
26     <tbody>
27         @if (items == null)
28         {
29             <tr><td>Loading...</td></tr>
30         }
31         else
32         {
33             foreach (var item in items)
34             {
35     <tr>
36         <td>@item.ReaInsId</td>
37         <td>@item.ReaAuditCd</td>
38         <td>@(item.ReaData.HasValue ? item.ReaData.Value.ToString("0.0") : "")</td>
39         <td>@item.ReaSequence</td>
40         <td>@item.ReaTemperature</td>
41         <td>@item.ReaHumidity</td>
42         <td>@item.ReaVoltage</td>
43         <td>@item.ReaRSSI</td>
44         <td><p><a class="btn btn-sm btn-outline-primary" href=@($"/readouts/edit/{item.ReaId}"
       ) role="button"><span class="oi oi-pencil" aria-hidden="true"></span></a></p></td>
45     </tr>
46             }
47         }
48     </tbody>
49 </table>
50
51 @code {
52
53     IEnumerable<TReadouts> items;
```

```
54      protected override async Task OnInitializedAsync()
55      {
56          items = await Task.Run(() => DataAccessService.TReadoutsSelectAll());
57      }
58  }
```

C. Network configuration scripts

C.1. Network address translation

Listing C.1. NAT Mapping

```
Add-NetNatStaticMapping -ExternalIPAddress "0.0.0.0/24" -ExternalPort 80 -Protocol TCP -InternalIPAddress "192.168.0.22" -InternalPort 80 -NatName NATNetwork

Add-NetNatStaticMapping -ExternalIPAddress "0.0.0.0/24" -ExternalPort 443 -Protocol TCP -InternalIPAddress "192.168.0.22" -InternalPort 443 -NatName NATNetwork

Add-NetNatStaticMapping -ExternalIPAddress "0.0.0.0/24" -ExternalPort 1883 -Protocol TCP -InternalIPAddress "192.168.0.23" -InternalPort 1883 -NatName NATNetwork
```

C.2. .NET configuration

Listing C.2. MSSQL RDBMS installation

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/config/ubuntu/18.04/mssql-server-2019.list)"
sudo apt-get update
sudo apt-get install -y mssql-server
sudo /opt/mssql/bin/mssql-conf setup
systemctl status mssql-server --no-pager
```

Listing C.3. MSSQL Client tools

```
sudo apt-get -y upgrade
curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list | sudo tee /etc/apt/sources.list.d/msprod.list
sudo apt-get install mssql-tools unixodbc-dev
```

Listing C.4. .NET Core installation

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
sudo apt-get update; \
```

```
  sudo apt-get install -y apt-transport-https && \
  sudo apt-get update && \
  sudo apt-get install -y dotnet-sdk-3.1
```

Listing C.5. Blazor demo for test purposes

```
dotnet new -i Microsoft.AspNetCore.Blazor.Templates::3.1.0-preview4.19579.2
dotnet new blazorwasm -o BlazorDemo
cd BlazorDemo
dotnet run
```

Listing C.6. Blazor start script

```
#!/bin/sh
dotnet run  --urls http://0.0.0.0:5005 --project ~/datacloud/Sources/DataCloud/DataCloud -v n
```

C.3. Gateway configuration

Listing C.7. Relayd configuration

```
/etc/relayd.conf
external_ip = "192.168.0.22"
forward_ms = "192.168.0.21"
localhost = "127.0.0.1"

table <forward_datacloud_table> { $forward_ms }
table <acme-challenge> { $localhost }

forward_datacloud_port = "5005"
table <forward_datacloud_table_local> { $localhost }
forward_datacloud_port_local = "http"

http protocol "https" {
    tls keypair "datacloud.atner.pl"
    pass request quick header "Host" value "datacloud.atner.pl" forward to <
    forward_datacloud_table>
    match response header remove "Server"
    match response header set "X-Frame-Options" value "SAMEORIGIN"
    match response header set "X-XSS-Protection" value "1; mode=block"
    match response header set "X-Content-Type-Options" value "nosniff"
    match response header set "Referrer-Policy" value "strict-origin"
#    match response header set "Content-Security-Policy" value "default-src 'self'"
    match response header set "Feature-Policy" value "accelerometer 'none'; camera 'none';
    geolocation 'none'; gyroscope 'none'; magnetometer 'none'; microphone 'none'; payment '
    none'; usb 'none'"
    tcp { nodelay, sack, socket buffer 65536, backlog 100 }
    block
}

http protocol "http" {
    pass request quick path "/.well-known/acme-challenge/*" forward to <acme-challenge>
    pass request quick header "Host" value "datacloud.atner.pl" forward to <
    forward_datacloud_table_local>
```

```
30      match response header remove "Server"
31      match response header set "X-Frame-Options" value "SAMEORIGIN"
32      match response header set "X-XSS-Protection" value "1; mode=block"
33      match response header set "X-Content-Type-Options" value "nosniff"
34      match response header set "Referrer-Policy" value "strict-origin"
35      match response header set "Content-Security-Policy" value "default-src 'self'"
36      match response header set "Feature-Policy" value "accelerometer 'none'; camera 'none';
        geolocation 'none'; gyroscope 'none'; magnetometer 'none'; microphone 'none'; payment '
        none'; usb 'none'"
37      tcp { nodelay, sack, socket buffer 65536, backlog 100 }
38      block
39  }
40
41  relay "https" {
42      listen on $external_ip port https tls
43      protocol "https"
44      forward to <forward_datacloud_table> port $forward_datacloud_port
45  }
46
47  relay "http" {
48      listen on $external_ip port http
49      protocol "http"
50      forward to <forward_datacloud_table_local> port $forward_datacloud_port_local
51  }
```

Listing C.8. Httpd configuration

```
1  localhost = "127.0.0.1"
2
3  types {
4      include "/usr/share/misc/mime.types"
5  }
6
7  server "default" {
8      listen on $localhost port http
9      location "/.well-known/acme-challenge/*" {
10         root "/acme"
11         request strip 2
12     }
13     location "*" {
14         block return 301 "https://$HTTP_HOST$REQUEST_URI"
15     }
16 }
```

Listing C.9. SSL certificate renewal script

```
1  authority letsencrypt {
2          api url "https://acme-v02.api.letsencrypt.org/directory"
3  #        api url "https://acme-staging-v02.api.letsencrypt.org/directory"
4          account key "/etc/acme/letsencrypt-privkey.pem"
5  }
6
7  domain datacloud.atner.pl {
8          #alternative names { www.datacloud.atner.pl }
```

```
9           domain key "/etc/ssl/private/datacloud.atner.pl.key"
10          domain certificate "/etc/ssl/datacloud.atner.pl.crt"
11          domain full chain certificate "/etc/ssl/datacloud.atner.pl.pem"
12          sign with letsencrypt
13  }
```

Listing C.10. Initial configuration and crontab

```
1  relayd -n
2  rcctl enable relayd
3  rcctl restart relayd
4
5  httpd -n
6  rcctl enable httpd
7  rcctl restart httpd
8
9  acme-client datacloud.atner.pl
10
11 crontab -e
12 27 5 * * * /usr/sbin/acme-client datacloud.atner.pl && /usr/sbin/relayctl reload
```

D. Schematics

In this appendix we show schematics of all devices described in this work as well as PCB visualisation and bill of materials.

Table D.1. Bill of materials of the first version of the device

Designator	Manufacturer	Part name	Price [$]
U3	Sensirion	SHT30-DIS-B10KS	1.407
1W+	BOOMELE	Header-Male-2.54-1x3	0.015
X1	SEIKO	SSP-T7-F 32.768KHz	0.413
U2	STMicroelectronics	STM32L072CBT6	7.736
C2,C3	SAMSUNG	CL10C120JB8NNNC 12pF	0.005
C5,C22,C23,C21,C1	YAGEO	CC0603KRX7R9BB104 100nF	0.002
R2,R3	UniOhm	0603WAF4701T5E 4.7K	0.001
USB	Jing	918-468K2029E50001	0.636
3-3.6V	Keystone	KEYS54 Battery 18350	0.69
R1,R5	UniOhm	0603WAF2000T5E 200	0.001
D2,D1	KENTO	0603White light	0.1
DU	Tech public	TPUSBLC6-2SC6	0.073
R6	UniOhm	0603WAF470KT5E 4.7	0.002
R4	Tyohm	RMC06031.5K1%N 1.5K	0.003
U1	Ai-Thinker	Ra-02	5.261

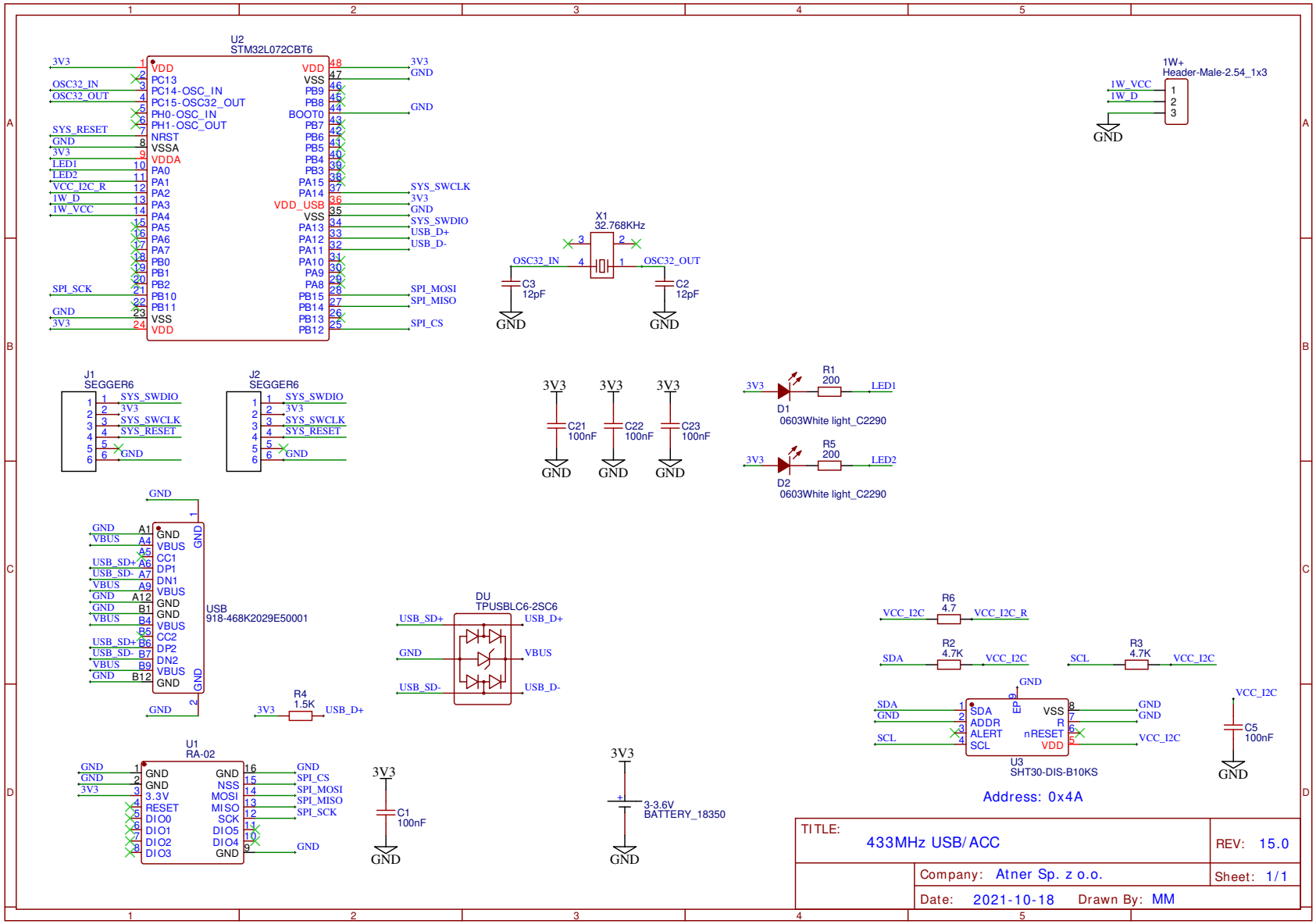

Figure D.1. Schematics of the first version of the device

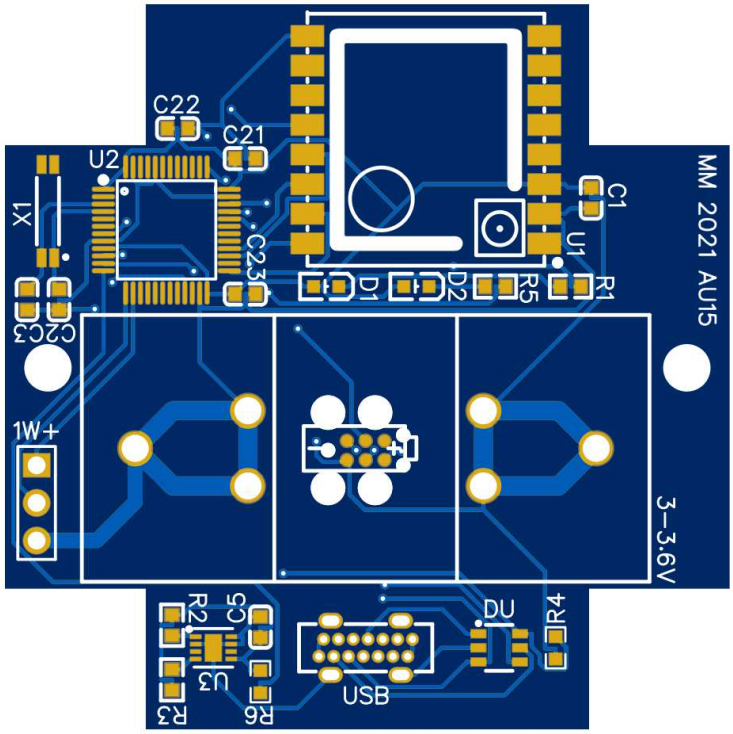

Figure D.2. Printed circuit board of the first version of the device

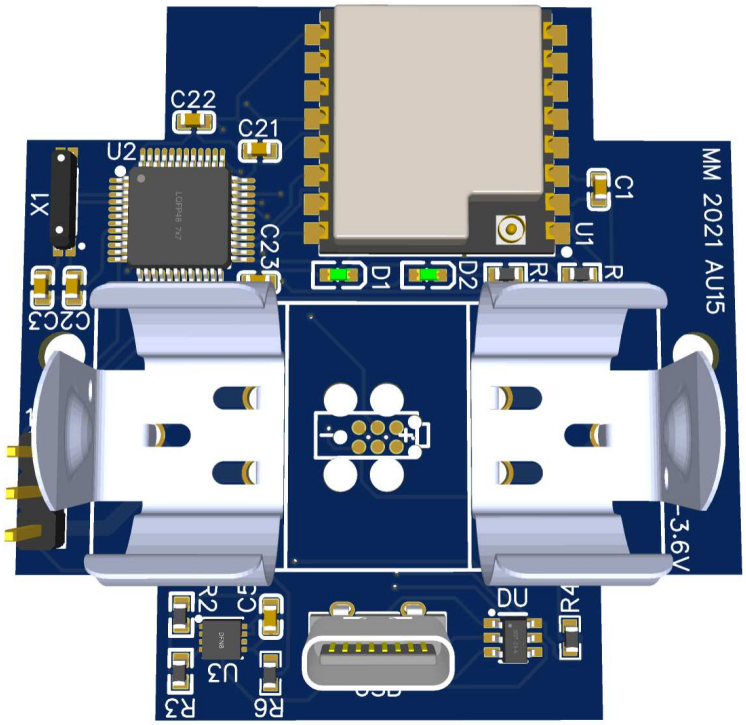

Figure D.3. Visualisation of the first version of the device

Table D.2. Bill of materials of the second version of the device

Designator	Manufacturer	Part name	Price [$]
U2	STMicroelectronics	STM32L151C8T6	3.59
X1	YXC	X322516MLB4SI 16MHz	0.085
X2	EPSON	Q13FC1350000400 32.768KHz	0.23
C4,C6,C2,C3	SAMSUNG	CL10C120JB8NNNC 12pF	0.005
R4	UniOhm	0603WAF1501T5E 1.5K	0.001
R7,R8	UniOhm	0603WAF5101T5E 5.1K	0.001
U3	Sensirion	SHT30-DIS-B10KS	1.407
C5,C22,C23,C21,C1	YAGEO	CC0603KRX7R9BB104 100nF	0.002
R2,R3	UniOhm	0603WAF4701T5E 4.7K	0.001
USB	Jing	918-468K2029E50001	0.636
3-3.6V	Keystone	KEYS54 Battery 18350	0.69
R1,R5	UniOhm	0603WAF2000T5E 200	0.001
D2,D1	KENTO	0603White light	0.1
DU	Tech public	TPUSBLC6-2SC6	0.073
R6	UniOhm	0603WAF470KT5E 4.7	0.002
U1	Ai-Thinker	Ra-02	5.261

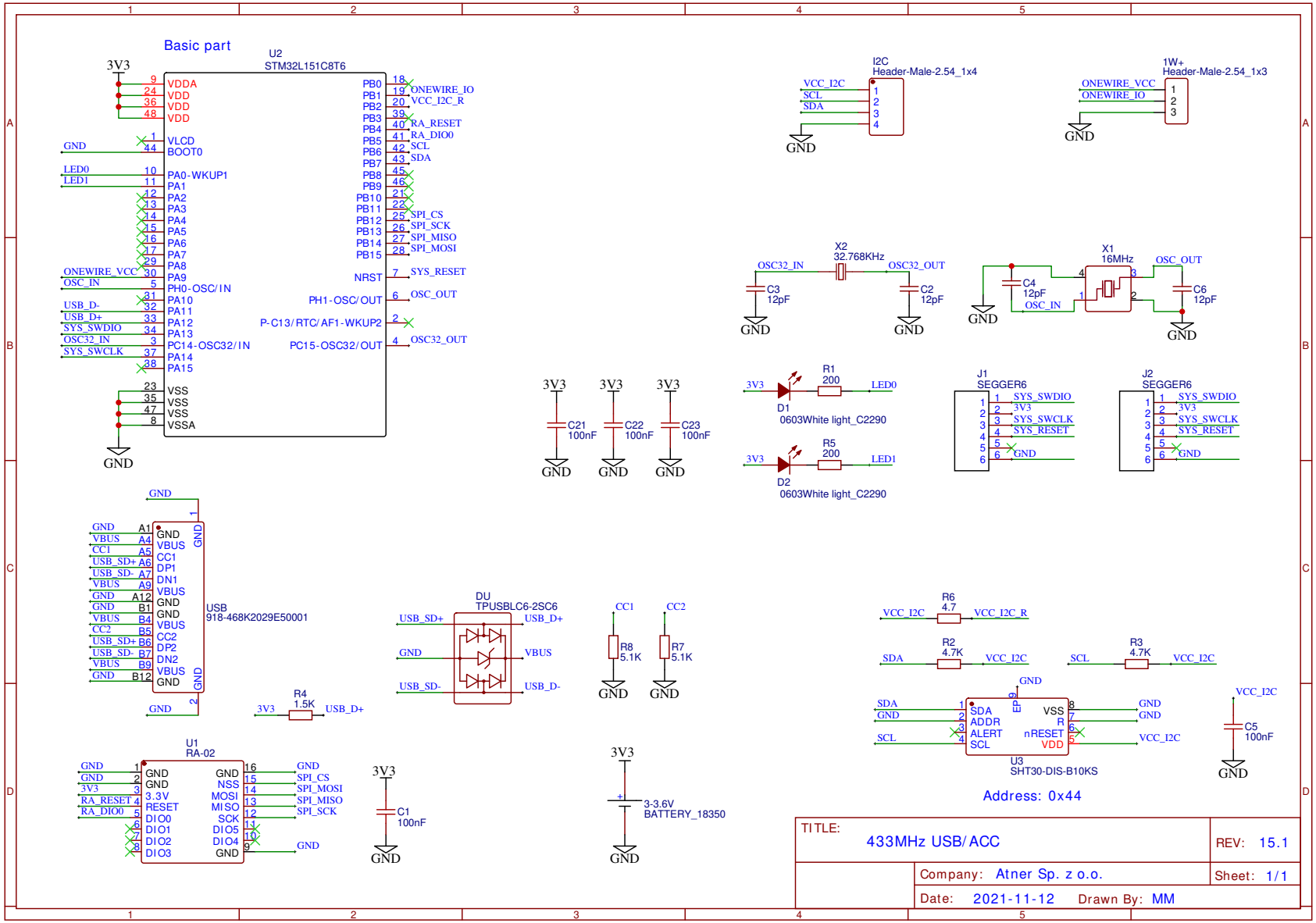

Figure D.4. Schematics of the second version of the device

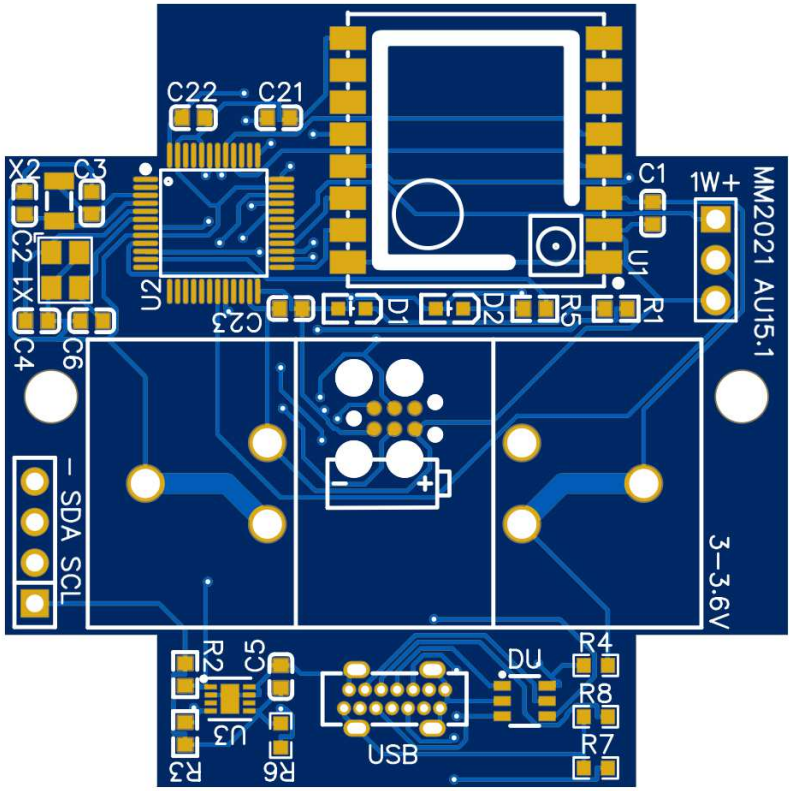

Figure D.5. Printed circuit board of the second version of the device

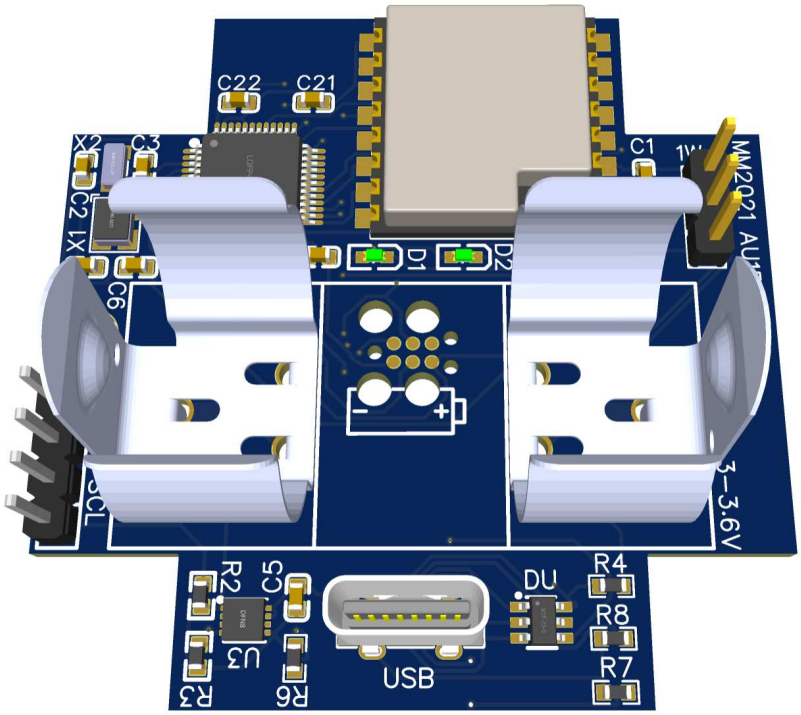

Figure D.6. Visualisation of the second version of the device

Table D.3. Bill of materials of the third version of the device

Designator	Manufacturer	Part name	Price [$]
U6	STMicroelectronics	SPV1050TTR	4.3
R24	UniOhm	0603WAF1005T5E 10M	0.001
R25	UniOhm	0603WAF1004T5E 1M	0.001
R26	UniOhm	0603WAF5104T5E 5.1M	0.001
R27,R19	UniOhm	0603WAF5103T5E 510K	0.001
C7	SAMSUNG	CL10A105KB8NNNC 1uF	0.003
L1	Coilank Int.	APS07A30M220 22uH	0.212
D3,D4	MDD	SS14	0.016
C11,C10,C5,C22, C23,C21,C1	YAGEO	CC0603KRX7R9BB104 100nF	0.002
U4	Vollgo	SX1278S47S+T-X1	5.402
Q1	AOS	AO3400A	0.131
R9,R10,R11, R12,R13,R14, R15,R16,R17	UniOhm	0603WAF1003T5E 100K	0.001
USBC	Korean Hroparts	TYPE-C-31-M-12	0.233
C8	SAMSUNG	CL31A476MPHNNNE 47uF	0.071
C9	FH	0603B103K500NT 10nF	0.001
R18	UniOhm	0603WAF2704T5E 2.7M	0.001
R20	UniOhm	0603WAF6204T5E 6.2M	0.001
U2	STMicroelectronics	STM32L151C8T6	3.59
X1	YXC	X322516MLB4SI 16MHz	0.085
X2	EPSON	Q13FC1350000400 32.768KHz	0.23
C4,C6,C2,C3	SAMSUNG	CL10C120JB8NNNC 12pF	0.005
R4	UniOhm	0603WAF1501T5E 1.5K	0.001
R7,R8	UniOhm	0603WAF5101T5E 5.1K	0.001
U3	Sensirion	SHT30-DIS-B10KS	1.407
R2,R3	UniOhm	0603WAF4701T5E 4.7K	0.001
R1,R5	UniOhm	0603WAF2000T5E 200	0.001
D2,D1	KENTO	0603White light	0.1
DU	Tech public	TPUSBLC6-2SC6	0.073
R6	UniOhm	0603WAF470KT5E 4.7	0.002
3-3.6V	Keystone	KEYS54 Battery 18350	0.69

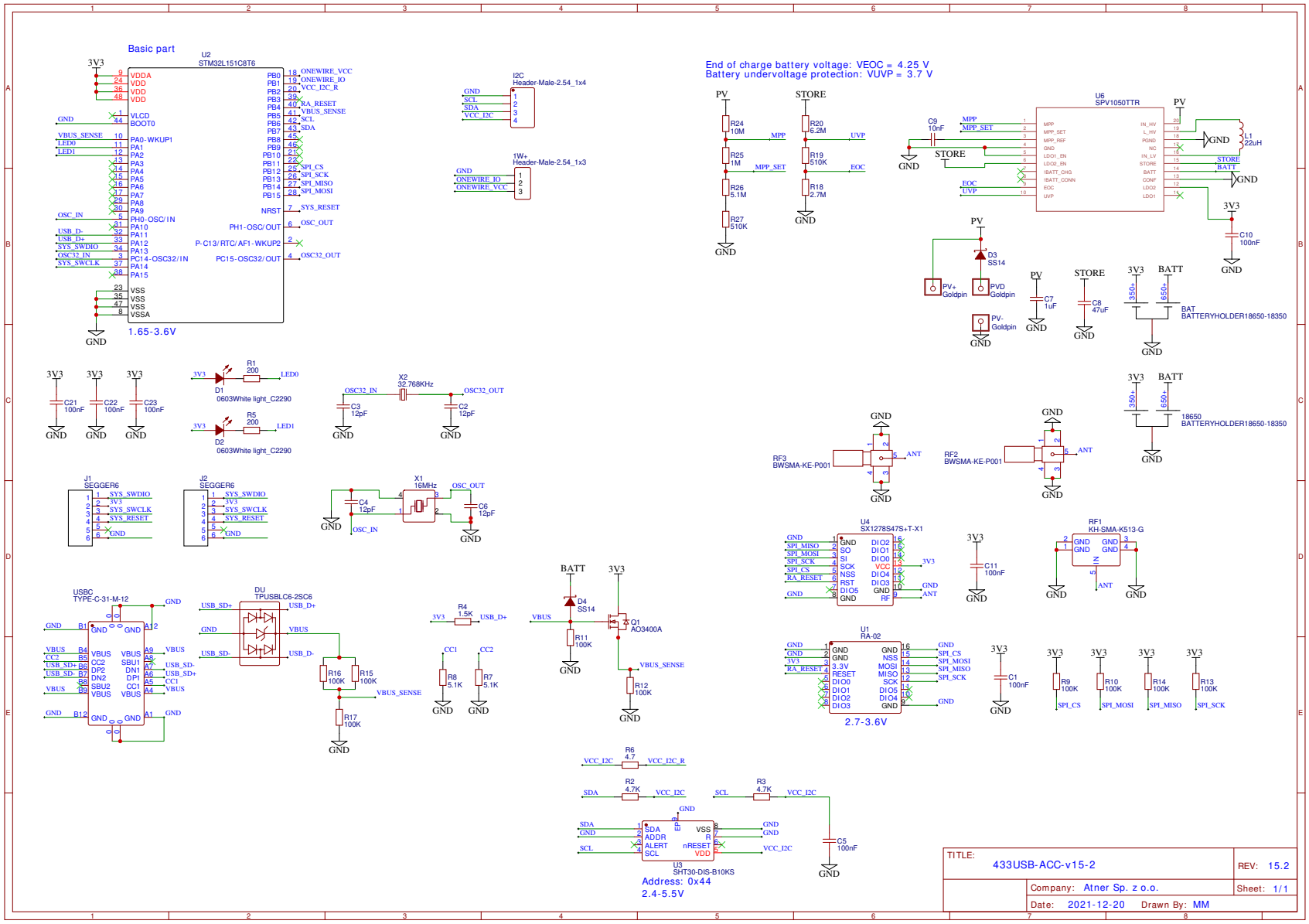

Figure D.7. Schematics of the third version of the device

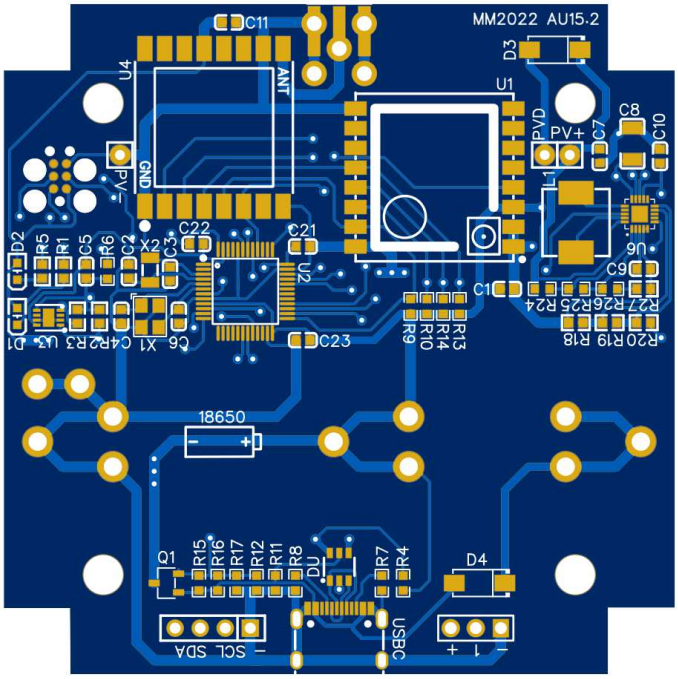

Figure D.8. Printed circuit board of the third version of the device

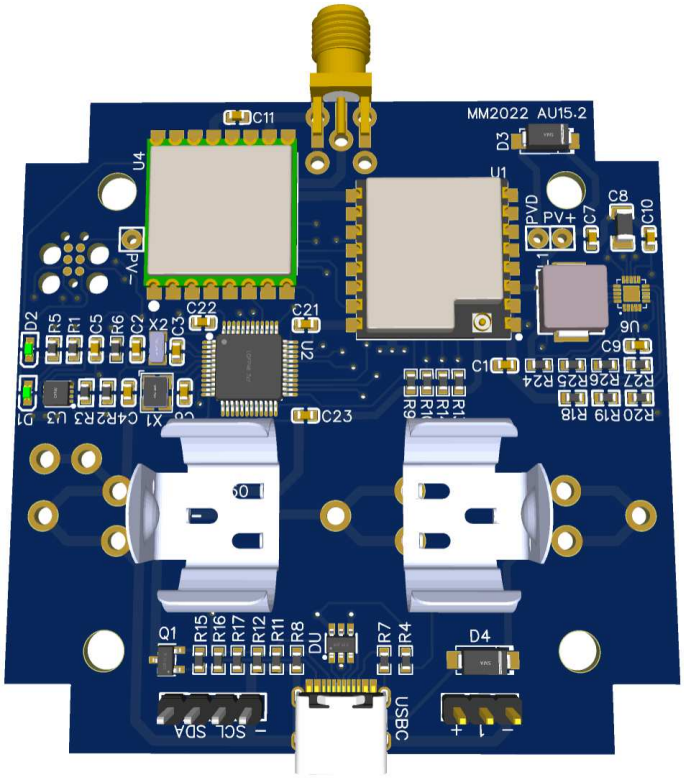

Figure D.9. Visualisation of the third version of the device

Table D.4. Bill of materials of the fourth version of the device

Designator	Manufacturer	Part name	Price [$]
USB	SOFNG	MC-311D	0.559
D7,D4,D5	ROHM Semicon	RB715FT106	0.118
D8,D9,D1,D3	SEMTECH	LL4148	0.007
R6,R10,R16,R20, R9,R11,R12	UniOhm	0603WAF1003T5E 100K	0.001
R17,R5	UniOhm	0603WAF2000T5E 200	0.001
D6	SEMTECH	ZMM5V6	0.018
R2,R3,R7,R8	UniOhm	0603WAF5101T5E 5.1K	0.001
Q6,Q9	AOS	AO3400A	0.131
R1,R13,R14, R15,R22,R23, R24,R25,R27	UniOhm	0603WAF1005T5E 10M	0.001
C12,C11,C5,C22, C23,C21,C1	YAGEO	CC0603KRX7R9BB104 100nF	0.002
C31,C32	SAMSUNG	CL05A105KA5NQNC 1uF	0.004
Q3	AOS	AO3401A	0.097
U5	MICROCHIP	MCP6541T-I/OT	1.851
U33	TOREX	XC6206P332MR	0.104
U4	Vollgo	SX1278S47S+T-X1	5.402
U2	STMicroelectronics	STM32L151C8T6	3.59
X1	YXC	X322516MLB4SI 16MHz	0.085
X2	EPSON	Q13FC1350000400 32.768KHz	0.23
C4,C6,C2,C3	SAMSUNG	CL10C120JB8NNNC 12pF	0.005
R4	UniOhm	0603WAF1501T5E 1.5K	0.001
U3	Sensirion	SHT30-DIS-B10KS	1.407
D2	KENTO	0603White light	0.1
3-3.6V	Keystone	KEYS54 Battery 18350	0.69

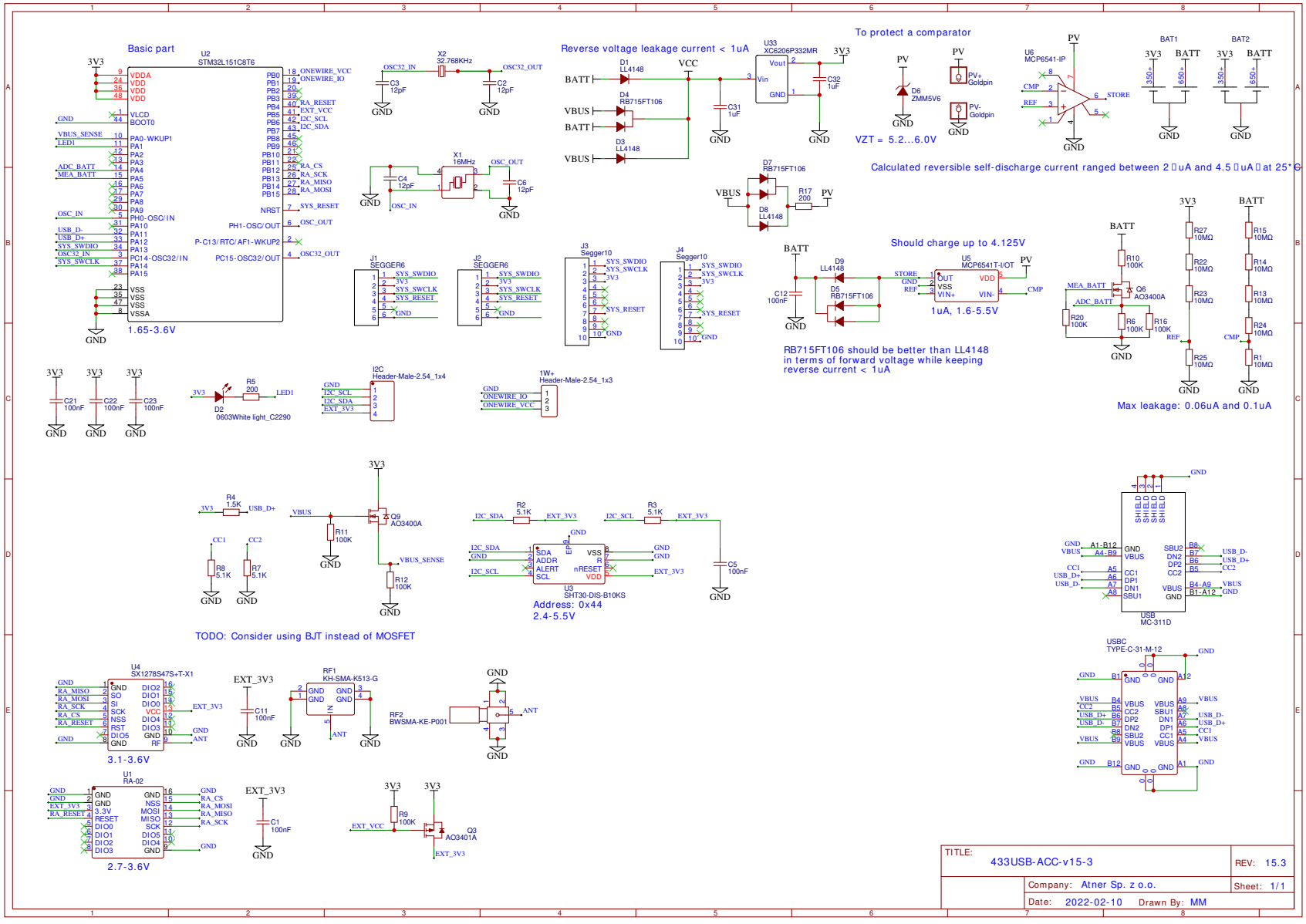

Figure D.10. Schematics of the fourth version of the device

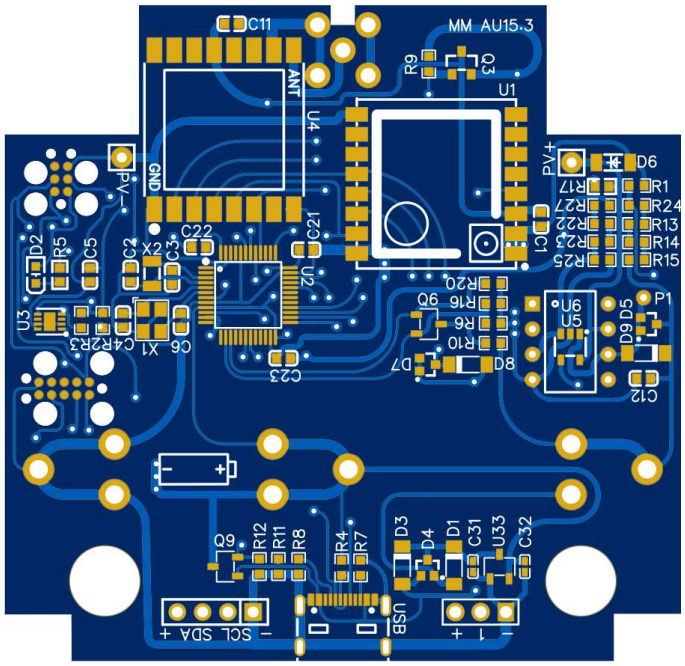

Figure D.11. Printed circuit board of the fourth version of the device

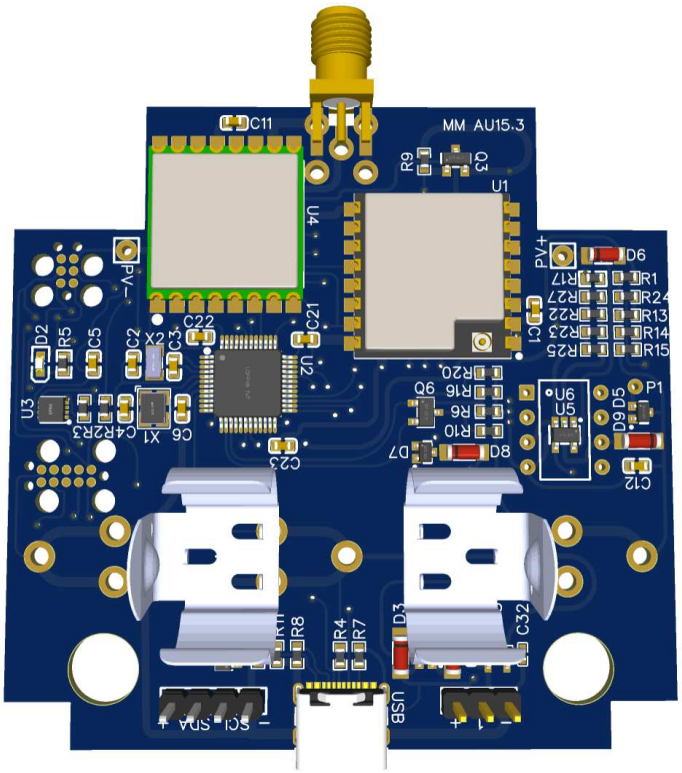

Figure D.12. Visualisation of the fourth version of the device

Table D.5. Bill of materials of the final version of the device

Designator	Manufacturer	Part name	Price [$]
BAT1	Keystone	KEYS54 Battery 18350	0.69
U10	AMS	CCS811B-JOPD500	7.831
U11	Bosch	BMP280	1.982
C9,C27,C12,C8,C5,C17-24,C1	SAMSUNG	CL05B104KO5NNNC 100nF	0.001
USB	SOFNG	MC-311D	0.559
U8	SEMTECH	SX1278IMLTRT 137MHz – 525MHz	3.39
R1,R29,R13-15,R19-26	UniOhm	0402WGF1005TCE 10M	0.001
X3	Yangxing	X503232MSB4SI 32MHz	0.237
L9	Sunlord	SDCL1005C12NJTDF 12nH	0.003
L8	Sunlord	SDCL1005C15NJTDF 15nH	0.003
L7	Sunlord	SDCL1005C18NJTDF 18nH	0.004
L1	Sunlord	SDCL1005C82NJTDF 82nH	0.003
C33,C14,C13	FH	0402CG470J500NT 47pF	0.001
C30,C28	FH	0402CG6R8C500NT 6.8pF	0.001
C29	FH	0402CG2R7C500NT 2.7pF	0.001
C26,C16,C15	FH	0402CG4R7C500NT 4.7pF	0.001
C25	FH	0402CG2R2C500NT 2.2pF	0.001
C11,C10,C7,C6,C4,C3,C2	FH	0402CG120J500NT 12pF	0.001
R20,R16,R12,R11,R10,R9,R6	UniOhm	0402WGF1003TCE 100k	0.001
R18,R17,R5	UniOhm	0402WGF2000TCE 200	0.001
R3,R2,R8,R7	UniOhm	0402WGF5101TCE 5.1K	0.001
R4	UniOhm	0402WGF1501TCE 1.5K	
Q4,Q6,Q9	AOS	AO3400A	0.131
Q1,Q3	AOS	AO3401A	0.097
D7,D4,D10,D8,D6,D3,D2	SEMTECH	LL4148	0.007
X2	EPSON	Q13FC1350000400 32.768KHz	0.23
X1	YXC	X322516MLB4SI 16MHz	0.085
U7	TI	CD40106BM96	0.369
U5	MICROCHIP	MCP6541T-I/OT	1.851
U3	Sensirion	SHT30-DIS-B10KS	1.407
U2	STM	F STM32L151C8T6	3.59
RF1	Kinghelm	KH-SMA-K513-G	0.52
U33	TOREX	XC6206P332MR	0.104
C32,C31	SAMSUNG	CL05A105KA5NQNC 1uF	0.004
D1	KENTO	0603White light	0.1

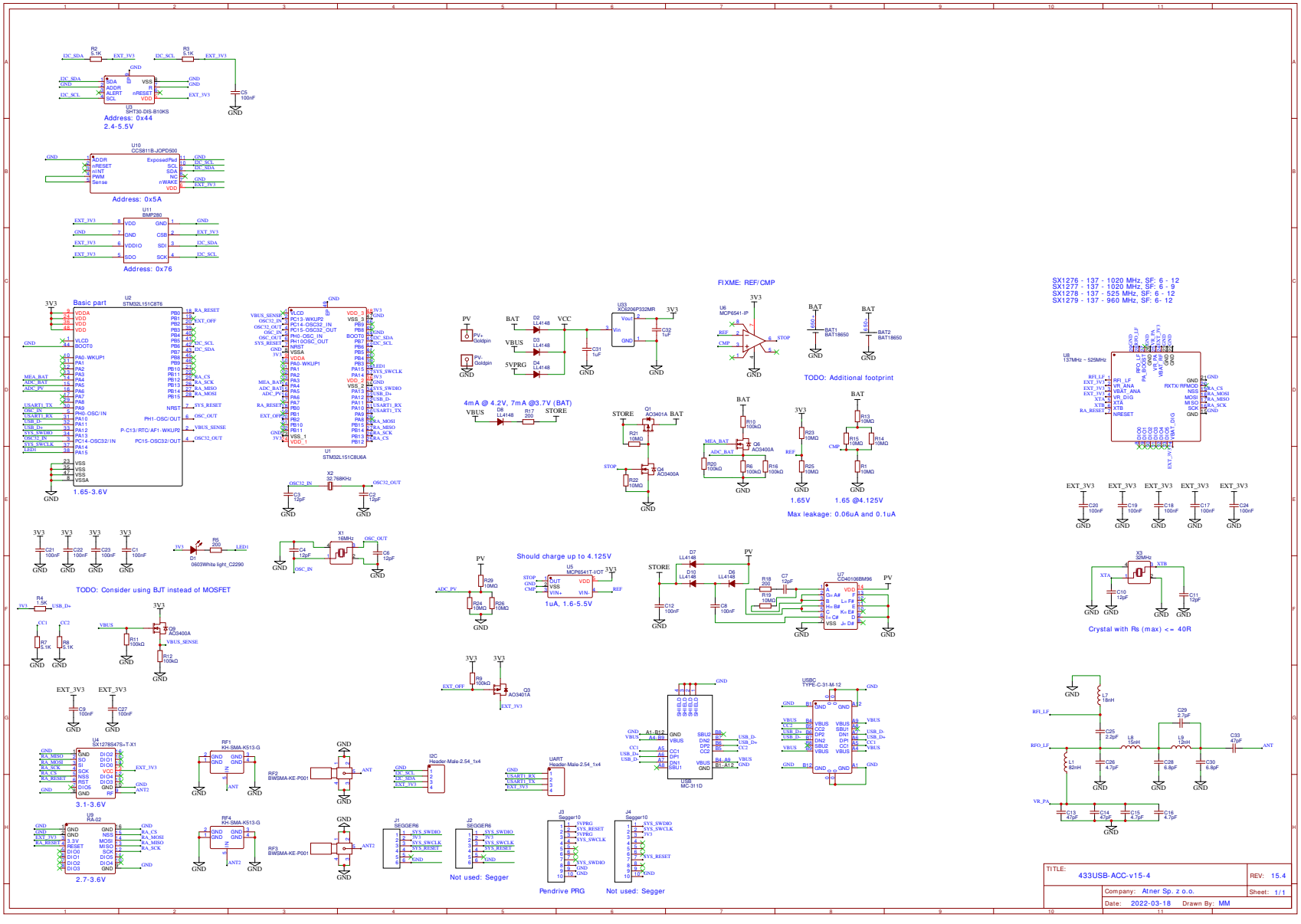

Figure D.13. Schematics of the fifth version of the device

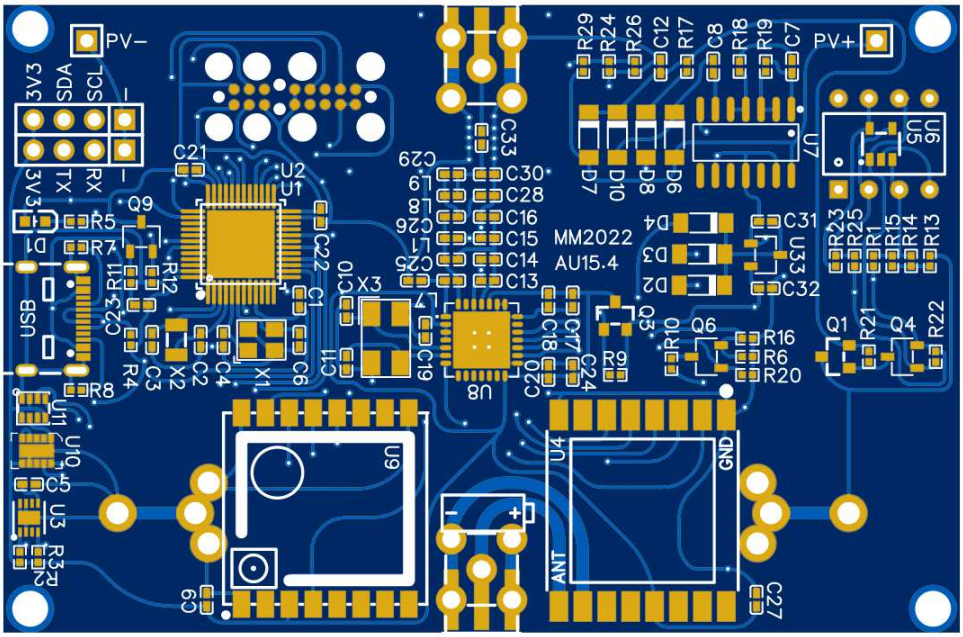

Figure D.14. Printed circuit board of the fifth version of the device

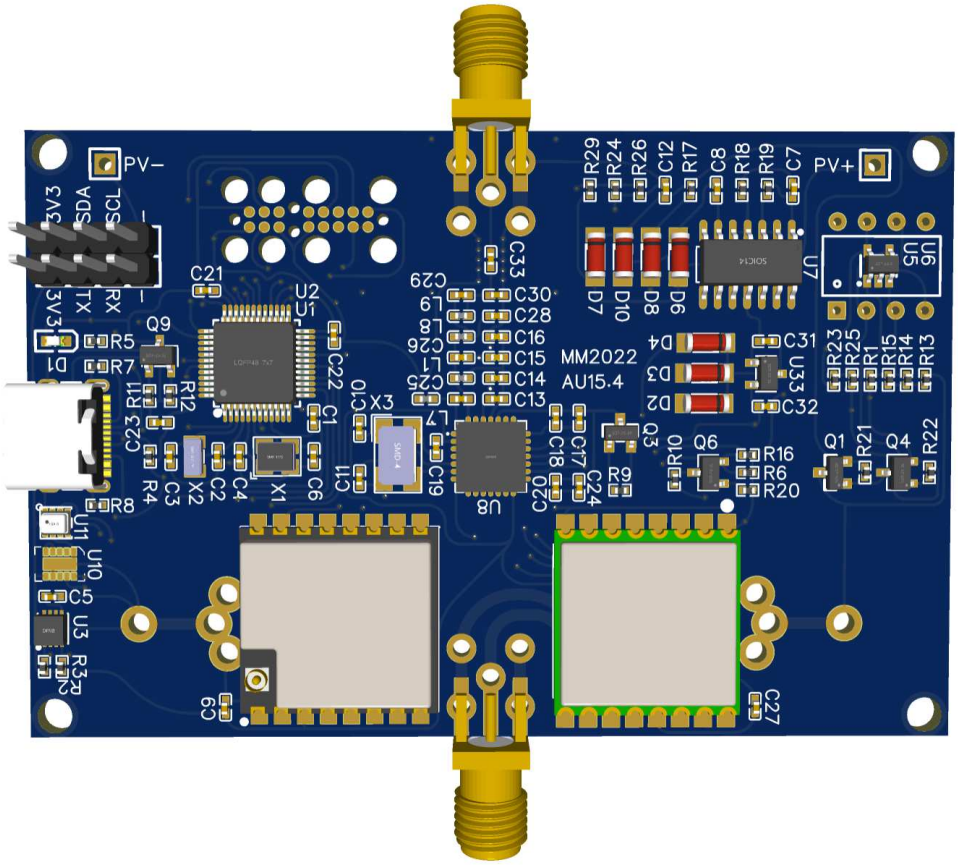

Figure D.15. Visualisation of the fifth version of the device

Table D.6. Bill of materials of the final version of the device

Designator	Manufacturer	Part name	Price [$]
BAT1	Keystone	KEYS54 Battery 18350	0.69
USB	SOFNG	MC-311D	0.559
D2,D3,D6,D8,D10, D4,D7	SEMTECH	LL4148	0.007
C1,C21,C22,C23,C8, C12,C27,C9,C5,C35,C34	SAMSUNG	CL05B104KO5NNNC 100nF	0.001
C31,C32	SAMSUNG	CL05A105KA5NQNC 1uF	0.004
U33	TOREX	XC6206P332MR	0.104
U2	STMicroelectronics	F STM32L151C8T6	3.59
U3	Sensirion	SHT30-DIS-B10KS	1.407
U5	MICROCHIP	MCP6541T-I/OT	1.851
U7	TI	CD40106BM96	0.369
X1	YXC	X322516MLB4SI 16MHz	0.085
X2	EPSON	Q13FC1350000400 32.768KHz	0.23
R4	UniOhm	0402WGF1501TCE 1.5K	
R7,R8,R2,R3	UniOhm	0402WGF5101TCE 5.1K	0.001
R5,R17,R18	UniOhm	0402WGF2000TCE 200	0.001
R6,R9,R10,R16,R20, R1,R23,R12,R11	UniOhm	0402WGF1003TCE 100k	0.001
R13,R19,R21,R22,R24, R26,R29	UniOhm	0402WGF1005TCE 10M	0.001
C2,C3,C4,C6,C7	FH	0402CG120J500NT 12pF	0.001
U4	Vollgo	SX1278S47S+T-X1	5.402
U11	Bosch	BMP280	1.982
U10	Ams AG	CCS811B-JOPD500	7.831
U12	Sensirion	SHT40-AD1B-R2	1.311
R15,R14	Uniroyal Elec	0402WGF1305TCE 13M	0.004
U15	Tech public	XC6206P152MR(1uA)	0.065
SW1	XKB Enterprise	TS-1187A-B-A-B	0.016
D1	KENTO	0603White light	
Q2,Q1	VISHAY	SI2301CDS-T1-GE3	0.052
Q3,Q4	CJ	2N7002	0.017
U8	MEMSIC	MXC4005XC	0.513

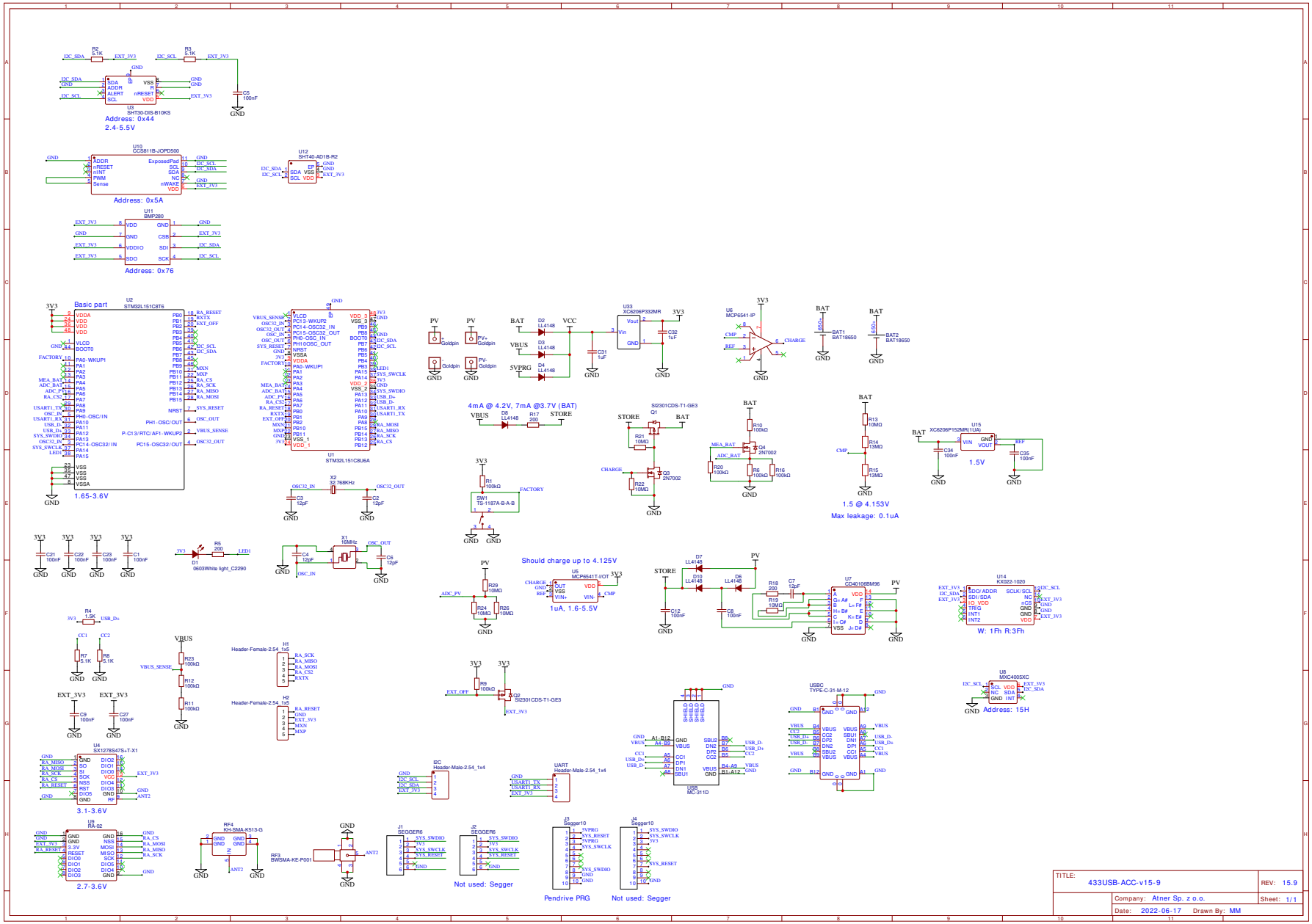

Figure D.16. Schematics of the sixth version of the device

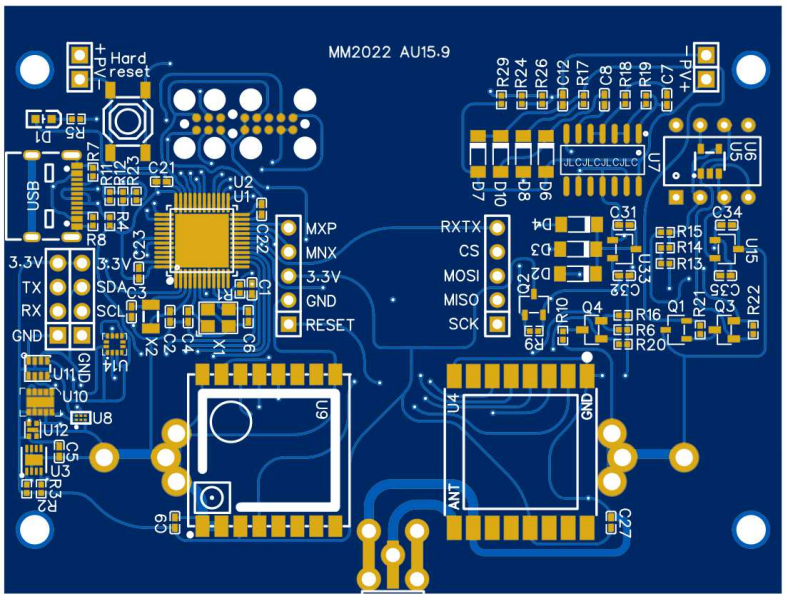

Figure D.17. Printed circuit board of the sixth version of the device

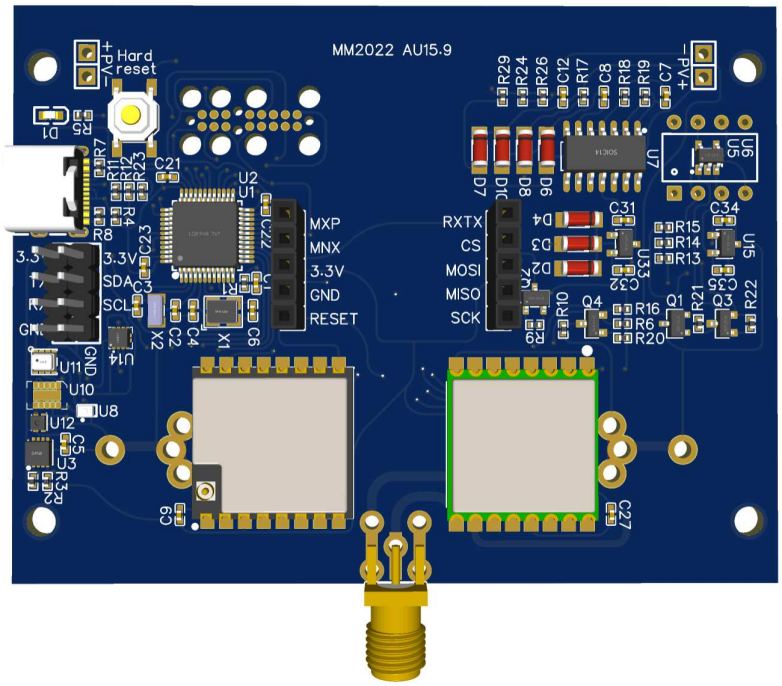

Figure D.18. Visualisation of the sixth version of the device

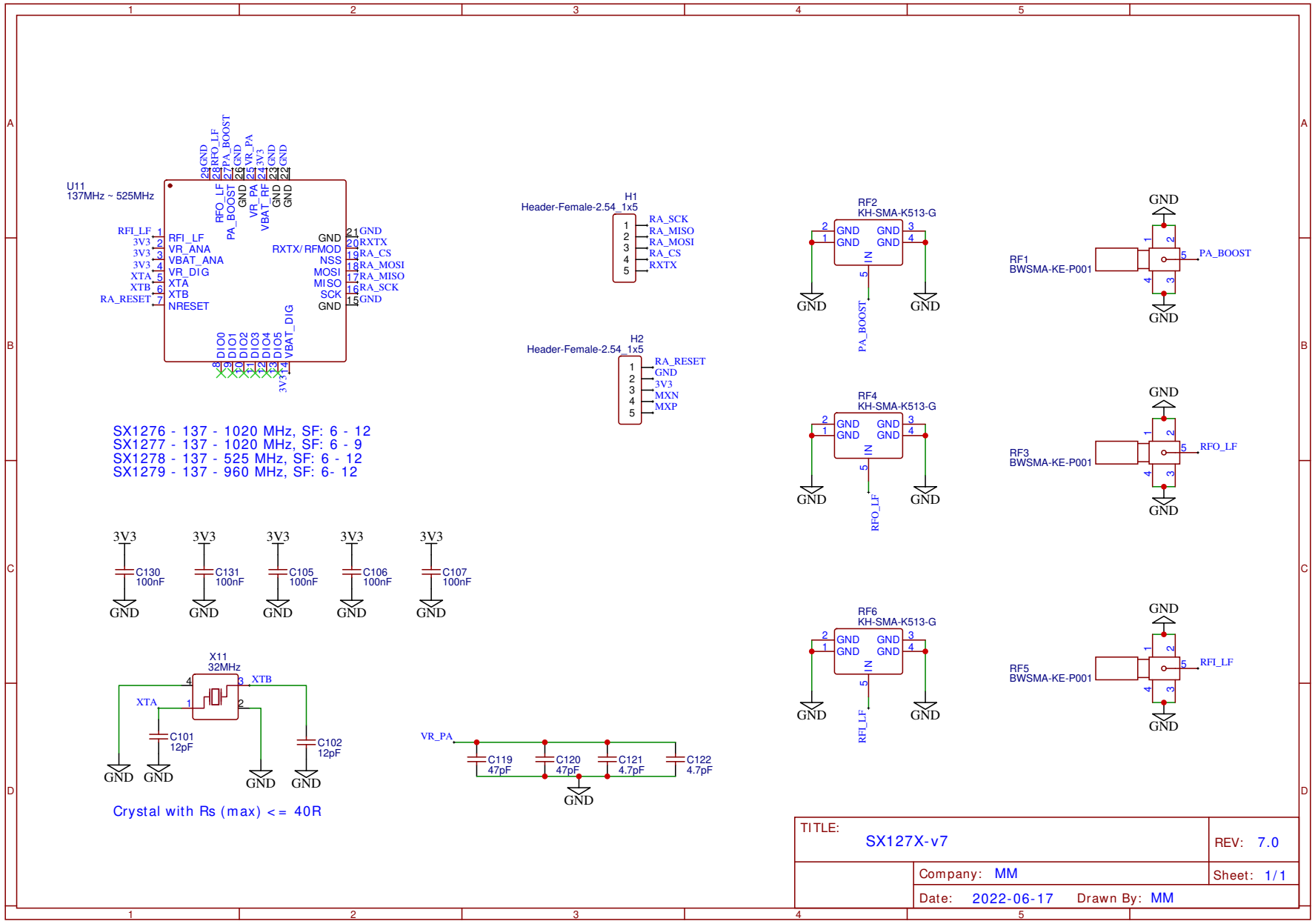

Figure D.19. Schematics of the radio module

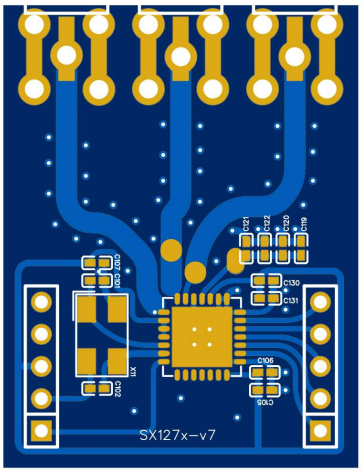

Figure D.20. Printed circuit board of the radio module

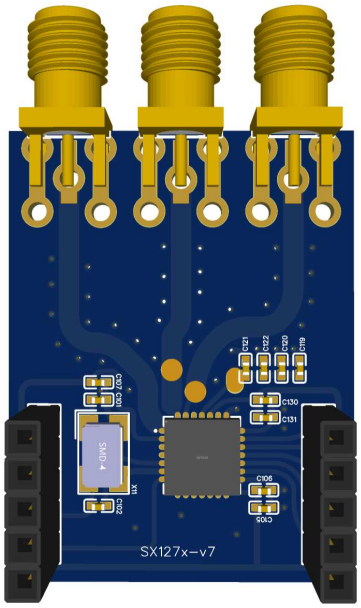

Figure D.21. Visualisation of the radio module

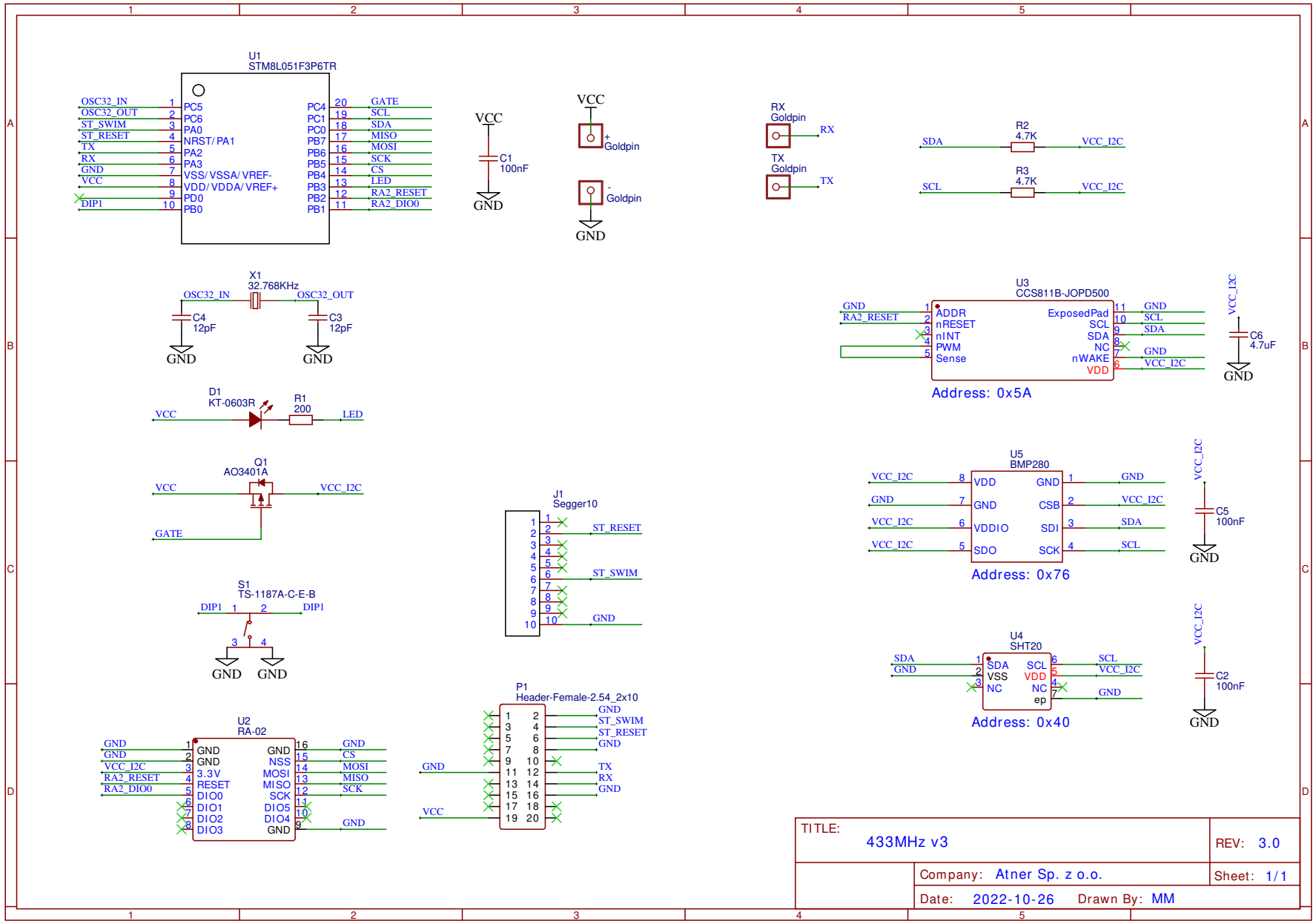

Figure D.22. Schematics of the seventh version of the device

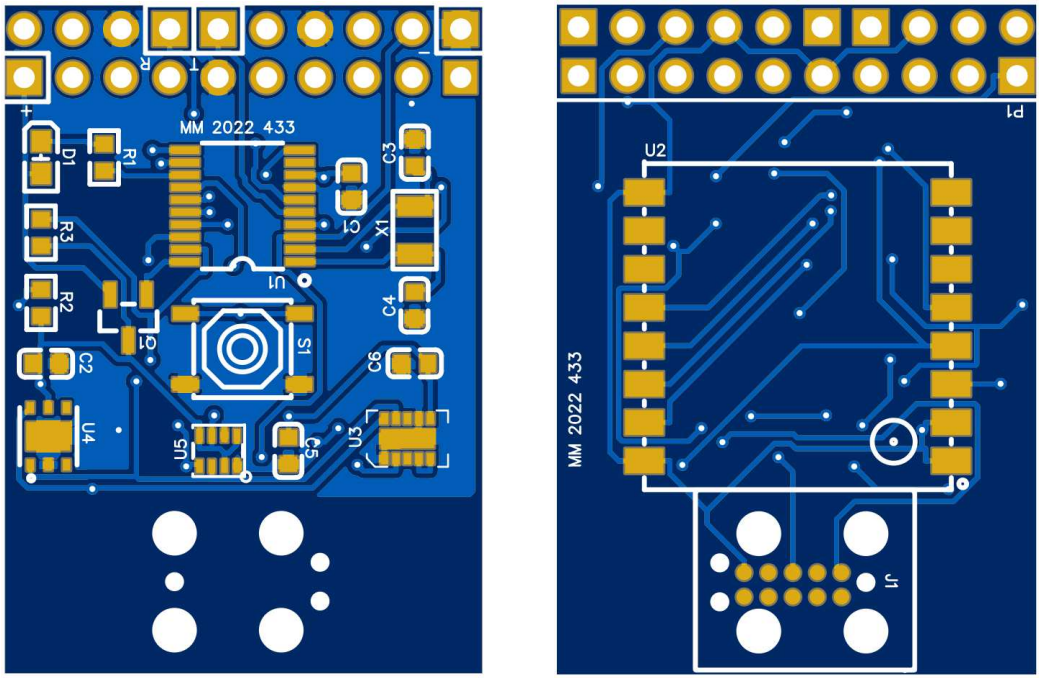

Figure D.23. Printed circuit board of the seventh version of the device

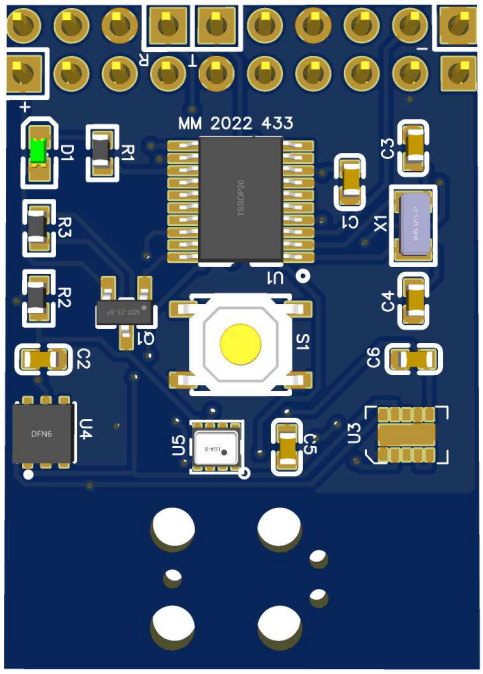

Figure D.24. Visualisation of the seventh version of the device

Bibliography

[1] *STM32L072x8, STM32L072xB, STM32L072xZ Datasheet*. (Accessed 2022-11-04). ST Microelectronics, Feb. 2019. URL: *https://www.st.com/resource/en/datasheet/stm32l072cz.pdf* (visited on 2022-11-04).

[2] *STM32L151x6/8/B, STM32L152x6/8/B Datasheet*. (Accessed 2022-11-04). ST Microelectronics, Apr. 2016. URL: *https://www.st.com/resource/en/datasheet/stm32l151c8.pdf* (visited on 2022-11-04).

[3] *MKW41Z/31Z/21Z Data Sheet*. MKW41Z512. Rev. 4. Accessed on 07/06/2019. NXP Semiconductors. Mar. 2018. URL: *https://www.nxp.com/docs/en/data-sheet/MKW41Z512.pdf* (visited on 2019-07-06).

[4] Mauri Kuorilehto et al. *Ultra-Low Energy Wireless Sensor Networks in Practice: Theory, Realization and Deployment*. Wiley Publishing, 2008. ISBN: 0470057866, 9780470057865.

[5] W. Yu et al. "A Survey on the Edge Computing for the Internet of Things". In: *IEEE Access* 6 (2018), pp. 6900–6919. ISSN: 2169-3536. DOI: *10.1109/ACCESS.2017.2778504*. URL: *https://ieeexplore.ieee.org/abstract/document/8123913*.

[6] Pascal Urard et al. "A self-powered IPv6 bidirectional wireless sensor & actuator network for indoor conditions". In: *2015 Symposium on VLSI Circuits (VLSI Circuits)* (2015), pp. C100–C101.

[7] Michał Markiewicz et al. "Software Controlled Low Cost Thermoelectric Energy Harvester for Ultra-Low Power Wireless Sensor Nodes". In: *IEEE Access* 8 (2020), pp. 38920–38930. DOI: *10.1109/ACCESS.2020.2975424*.

[8] Karunakar Pothuganti and Anusha Chitneni. "A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi". In: 4 (Sept. 2014), pp. 655–662.

[9] *SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver Datasheet*. (Accessed 2022-11-04). Semtech, May 2020. URL: *https://semtech.file.force.com/sfc/dist/version/download/?oid=00DE0000000JelG&ids=0682R000006TQEPQA4&d=%2Fa%2F2R0000001Rbr%2F6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE&operationContext=DELIVERY&asPdf=true* (visited on 2022-11-30).

[10] *LoRa Modulation Basics*. (Accessed 2022-11-30). Semtech, May 2015. URL: *https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf* (visited on 2022-11-30).

[11] Cengiz Camci. "Temperature and Heat Transfer Measurements". In: Dec. 2010, pp. 1432–1439. ISBN: ISBN: 978-0-470-75440-5 9780470686652. DOI: *10.1002/9780470686652.eae077*.

[12] *SHT40 Datasheet*. (Accessed 2022-12-03). Sensirion, Apr. 2022. URL: *https://sensirion.com/media/documents/33FD6951/624C4357/Datasheet_SHT4x.pdf* (visited on 2022-12-03).

[13] *AN1248 Application note*. (Accessed 2022-12-03). Analog Devices, Sept. 2015. URL: *https://www.analog.com/media/en/technical-documentation/application-notes/an-1248.pdf* (visited on 2022-12-03).

[14] *AN1159 Application note*. (Accessed 2022-12-03). Analog Devices, Sept. 2012. URL: *https://www.analog.com/media/en/technical-documentation/application-notes/an-1159.pdf* (visited on 2022-12-03).

[15] *AN1248 Application note*. (Accessed 2022-12-03). Energy Micro AS, Aug. 2011. URL: *https://www.digikey.dk/Site/Global/Layouts/DownloadPdf.ashx?pdfUrl=87B0C05A654F415EAB11D6D7A896553C* (visited on 2022-12-03).

[16] *MCP6541/1R/1U/2/3/4 Push-Pull Output Sub-Microamp Comparators Datasheet*. (Accessed 2022-11-05). Microchip, Feb. 2020. URL: *https://ww1.microchip.com/downloads/en/DeviceDoc/MCP6541%20Output%20SubMicroamp%20Comparators%2020001696K.pdf* (visited on 2022-11-05).

[17] *Raspberry Pi 4 Computer Datasheet*. (Accessed 2022-11-24). Raspberry Pi Trading Ltd., June 2019. URL: *https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf* (visited on 2022-11-24).

[18] *Ra-02 LoRa Product Specification V1.1*. (Accessed 2022-11-04). AI-Thinker, 2017. URL: *https://docs.ai-thinker.com/_media/lora/docs/c048ps01a1_ra-02_product_specification_v1.1.pdf* (visited on 2022-11-04).

[19] *SHT3x-DIS Datasheet Humidity and Temperature Sensor*. (Accessed 2022-11-04). Sensirion, 2017. URL: *https://sensirion.com/media/documents/213E6A3B/61641DC3/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital.pdf* (visited on 2022-11-04).

[20] *SPV1050 Datasheet – Ultralow power energy harvester and battery charger*. (Accessed 2022-11-05). ST Microelectronics, Jan. 2022. URL: *https://www.st.com/resource/en/datasheet/spv1050.pdf* (visited on 2022-11-05).

[21] *Amorton Amorphous Silicon Solar Cells*. (Accessed 2022-11-05). Panasonic, Mar. 2018. URL: *https://www.tme.eu/Document/b27698140f05f628e0174b0bed558abb/Panasonic%20Amorton%20solar-EN.pdf* (visited on 2022-11-05).

[22] *STM8L051F3 Datasheet*. (Accessed 2022-11-23). ST Microelectronics, Sept. 2018. URL: *https://www.st.com/resource/en/datasheet/stm8l051f3.pdf* (visited on 2022-11-23).

[23] *Power Profiler Kit II User Guide*. (Accessed 2022-11-04). Nordic Semiconductor, Aug. 2022. URL: *https://infocenter.nordicsemi.com/pdf/PPK2_User_Guide_v1.0.1.pdf* (visited on 2022-11-22).