BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2022

Bc. Vladimír Pokorný

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF POWER ELECTRICAL AND ELECTRONIC ENGINEERING

ÚSTAV VÝKONOVÉ ELEKTROTECHNIKY A ELEKTRONIKY

MEASURING TEMPERATURES IN THE ROTOR OF AN ELECTRICAL MACHINE

MĚŘENÍ TEPLOT V ROTORU ELEKTRICKÉHO STROJE

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Bc. Vladimír Pokorný
SUPERVISOR VEDOUCÍ PRÁCE	Ing. Jan Knobloch, Ph.D.

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study program **Power Electrical and Electronic Engineering**

Department of Power Electrical and Electronic Engineering

Student:	Bc. Vladimír Pokorný	***ID:***	200734
Year of study:	2	***Academic year:***	2021/22

TITLE OF THESIS:

Measuring temperatures in the rotor of an electrical machine

INSTRUCTION:

1. Make specific circuit diagrams of the functional sample.
2. Specify the mechanical design of the functional sample and design the circuit boards.
3. Design the control software parts.
4. Commission the functional sample and verify its functionality.

The work is carried out in collaboration with JKU - Johannes Kepler Universität Linz and the LCM - Linz Center of Mechatronics.

RECOMMENDED LITERATURE:

[1] ĎAĎO, Stanislav a Marcel KREIDL. Senzory a měřicí obvody. 2. vyd. Praha: Vydavatelství ČVUT, 1999, 315 s. ISBN 80-01-02057-6.

[2] KREIDL, Marcel. Měření teploty: senzory a měřicí obvody. Praha: BEN, 2005, 239 s. ISBN 80-7300-145-4.

[3] HUANG, Albert S a Larry RUDOLPH. Bluetooth essentials for programmers. New York, NY: Cambridge University Press, 2007, x, 198 s. : il. ; 24 cm. ISBN 978-0-521-70375-8.

Date of project specification:	7.2.2022	***Deadline for submission:***	22.5.2022

Supervisor: Ing. Jan Knobloch, Ph.D.

doc. Ing. Ondřej Vítek, Ph.D.
Chair of study program board

WARNING:

The author of the Master's Thesis claims that by creating this thesis he/she did not infringe the rights of third persons and the personal and/or property rights of third persons were not subjected to derogatory treatment. The author is fully aware of the legal consequences of an infringement of provisions as per Section 11 and following of Act No 121/2000 Coll. on copyright and rights related to copyright and on amendments to some other laws (the Copyright Act) in the wording of subsequent directives including the possible criminal consequences as resulting from provisions of Part 2, Chapter VI, Article 4 of Criminal Code 40/2009 Coll.

ABSTRACT

This Master's thesis deals with the problem of direct temperature measurement in the rotor of an electric motor. The main part of the thesis is the design of a high-end device for rotor temperature monitoring by a direct method. The designated temperature monitoring system is a 4 channel system. It uses PT100 or PT1000 resistance temperature detectors for measurement. Due to the used specialised MAX31865 circuit, the temperatures can be measured by 2, 3 and 4-wire configuration. Also, RTD sensors of different materials and nominal values can be used. This makes the device suitable for application in laboratory measurements. The experimental measurements revealed an average accuracy of 0.8°C from range $+20$ to $+200$°C

The power supply of the thermometer is provided by a small battery cell of $1/2$ AA size and a voltage of 3.6 V. The measured data is sent using the standardised Bluetooth Low Energy protocol so that the measuring system can be connected to a PC as well as to a laptop or smartphone with the appropriate client applications. No special receiver is not required.

The system as a whole is characterised by high noise immunity, low power consumption, high accuracy and mechanical robustness, which has been measured up to a speed of 6500 RPM. The paper concludes with practical measurements and the appendices contain documentation as well as a product brief in the appendix A.

KEYWORDS

Resistance Temperature Detector, PT100, Temperature Measuring, Bluetooth Low Energy, Motor Rotor Temperature, nRF52833, MAX31865

ABSTRAKT

Tato práce se zabývá problematikou měření teplot v rotoru elektrického stroje. Těžištěm práce je návrh, výroba a testování čtyřkanálového bezdrátového měřicího systému, který je primárně určen pro měření teplot v rotoru elektrického stroje.

První kapitola této práce – *Resistance temperature detectors* přibližuje problematiku použití odporových senzorů teploty, jejich normy, tolerance a měřicí obvody, které slouží pro měření odporu, respektive teploty.

Druhá kapitola – *SPI* vysvětluje základy symetrické sériové komunikační sběrnice SPI, která je v navrhovaném měřicím systému využita pro komunikaci mikrokontroléru s teplotními čidly.

Ve třetí kapitole *Bluetooth* autor vysvětluje komunikační standard Bluetooth Low Energy (BLE), díky kterému navrhované zařízení komunikuje a odesílá naměřená data ostatním zařízením v okolí. V úvodu této kapitoly jsou uvedeny základní parametry BLE a jeho výhody a nevýhody. Závěr této kapitoly pak rozebírá strukturu vrstev BLE.

Následuje obsáhlá čtvrtá kapitola *Hardware design*, kde je detailně popsán hardwarový návrh bezdrátového teploměru. Začátek této kapitoly je věnován napájecímu zdroji, který je v tomto případě složen z 3,6 V baterie velikosti 1/2 AA a lineárního napěťového regulátoru, který se stará o stabilizaci napájecího napětí. Následuje popis měřicího integrovaného obvodu MAX31865, který byl vybrán pro měření teploty díky jeho vysoké přesnosti a velkému měřicímu rozsahu. Aby se ušetřila elektrická energie v době nečinnosti zařízení, každý ze čtyř senzorů je vzdáleně zapínán a vypínán povelem mikrokontroléru. Posledním bodem této kapitoly je návrh desky plošných spojů s ohledem na vyvážení komponent, aby nedocházelo k nežádoucím vibracím vlivem vysokých otáček rotoru. Měřicí systém se bude otáčet společně s rotorem motoru, se kterým je pevně spojen. Pevné spojení dovoluje měřit teploty kontaktní metodou.

Pátá – podobně obsáhlá – kapitola *Software design* je zaměřena na popis firmware měřicího systému. První část kapitoly se zaměřuje na konfiguraci a inicializaci SPI sběrnice, nastavení komunikačních pinů a komunikační frekvence. Dále jsou popsány základní funkce pro čtení a zapisování registrů, které jsou využity pro řízení integrovaného obvodu MAX31865. Následuje popis implementace BLE do měřicího systému a popis hlavní smyčky programu (čtení teplot a vysílání naměřených dat pomocí BLE). Závěr této kapitoly se zabývá klientskou aplikací, která byla nad rámec zadání vytvořena a slouží pro příjem a ukládání dat z měřicího systému.

Závěrečná – šestá kapitola – *Device commissioning* popisuje výrobu, oživení a naprogramování výrobku. Měřicí systém byl otestován při 6500 ot/min a při měření kalibrační křivky byla zjištěna průměrná odchylka $+0,8$ % v rozsahu $+20°$C až $+200°$C. Přílohy na konci práce obsahují výrobní podklady. Produktový list výrobku lze nalézt v příloze A.

KLÍČOVÁ SLOVA

Odporové teplotní senzory, PT100, Měření teploty, Bluetooth Low Energy, Teplota v rotoru motoru, nRF52833, MAX31865

Typeset by the thesis package, version 4.07; http://latex.feec.vutbr.cz

Author's Declaration

Author:	Bc. Vladimír Pokorný
Author's ID:	200734
Paper type:	Master's Thesis
Academic year:	2021/22
Topic:	Measuring temperatures in the rotor of an electrical machine

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno . . . .

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank the advisor of my thesis, Ing. Jan Knobloch, Ph.D. for his long calls when we together solved a problem with the code, SPI and other problems with hardware. I would also like to thank DI Peter Dirnberger, who helped me to arrange many consultations with scientists at the JKU in Linz: DI Sebastian Außerwöger helped me with the PCB design, DI Leander Hörmann has been a great support for me in the Bluetooth implementation, Technicians Idil and Gerhard supported me the ordering components and drilling holes in tested motor. Mr. Ritter lent me his office and debugger.

At this point I would like to thank my parents for supporting me in my studies and even though it was often not easy I am very grateful to you! Thanks, Mom, for always taking care of me but now it's up to me.

I would like to thank Servotex company, where I was able to spend two days with my father producing this device. Thanks, Dad, for your patience and help during this days!

And here I would like to thank my colleagues in study: Pepa, Martin and Víťa thank you for Austria trips, dinners, tons of coffee cups and midnight talks. Also I would like thank to all of my friend who helped me in any way, whether it was a valuable advice or just a short phone call that pulled me back to reality!

It was an unforgettable experience, and at this moment I'm grateful that I was able to elaborate this thesis in four months into the form that you are now holding in your hands (or reading on a screen).

This work has been supported by the COMET-K2 Center of the Linz Center of Mechatronics (LCM) funded by the Austrian federal government and the federal state of Upper Austria.

And that is not all, this work was also supported by Scholarship AKTION financed by Federal Ministry of Education, Science and Research (BMBWF), awarded by ÖAD – Austria's Agency for Education and Internationalisation.

Contents

List of Figures

List of Tables

Listings

Introduction

The importance of correct motor thermal models has been discussed in numerous papers and publications. The knowledge of the thermal model is important for predicting the lifetime, maintenance intervals and reliability of a given motor. Each thermal model should be always verified in a real direct measurement.

While the stator motor temperature is easy to obtain, for many decades it has been a big challenge to measure the temperatures in the motor rotor. There are various possibilities to achieve this: one can use an IR sensor or thermal imaging camera, and another uses phosphor thermometry. All of these methods measure the temperatures only at the rotor surface. The temperaure information in the winding or under the rotor magnets is not available. In order to know the exact temperature at specific rotor locations, we need to place some thermometer on the rotor shaft that will measure the temperature inside the rotor with sufficient accuracy and the necessary fine sampling.

One such a device can be a thermometer that is placed on the motor rotor shaft, is powered by a battery, and measures and records temperatures on a storage medium. However, in this case, the evaluation of the temperatures occurs only after the measurements have been taken. The measurement data must be downloaded from the thermometer. If we need to evaluate the measured temperatures in real time, the measured data must be transmited from the thermometer immediately. There are many radio bands and communication protocols, differing in carrier frequency, interference immunity and power consumption. Since the designated thermometer will be mounted on a motor rotor and will be battery powered, Bluetooth Low Energy would be a suitable standard.

The design, development and production of a wireless thermometer for rotor temperature measurement is a challenge due to the high motor speed, high temperatures that are in the electric motor and the electromagnetic interference that is presented around the running electrical machine is another challenge. All of these together form a barrier that many scientific teams have not tried to overcome. This work tries to overcome this barrier and furthermore break it down so that others can measure real rotor temperatures of the electric motor simply.

1 Resistance temperature detectors

This chapter aims to introduce temperature measurement using Resistance Temperature Detectors (RTDs). First, the RTD sensors themselves are discussed. Next, industry standards and tolerances are introduced, and the chapter concludes with a discussion of the different types of sensor wiring and the effect of wiring on measurement accuracy.

1.1 Measuring Temperature

Temperature is one of the few physical quantities that cannot be measured directly. The measurement is taken using other physical quantities, and thus this measurement is indirect. It is important to note that a thermometer measures the temperature of its own sensor. So the sensor has to have a small thermal capacity, and the thermal resistance between the sensor and the measured place has to be small [1].

The thermometers could be divided into two groups:

1. Thermometers converting the temperature to electrical quantities
   - thermocouples, thermistors or RTD sensors and others
2. Thermometers converting the temperature to non-electrical quantities
   - thermal imaging cameras, bi-metals, liquid thermometers and others

1.2 Resistance temperature detectors

Resistance temperature detectors, or RTDs, are sensors used to measure temperature. RTDs are resistive elements that change their resistance over temperature. This dependence resistance on temperature is well characterised, and therefore they are used to make precision temperature measurements. It is typical to achieve accuracies of well under 0.1°C. [2]

The principle of RTDs is the temperature dependence of the metal resistance. For metals, the only temperature-dependent parameter is the electron relaxation time. For temperatures in the small temperature range from 0°C to +100°C, a linear relationship can be used with a small deviation [1]:

$$R_\vartheta = R_0(1 + \alpha\vartheta), \tag{1.1}$$

where the R_0 is the resistance of the metal sensor in 0°C and α is a temperature coefficient of resistance. Mean value of α could be expressed as:

$$\alpha = \frac{R_{100} - R_0}{100 \cdot R_0}, \tag{1.2}$$

where R_{100} is resistance at temperature 100°C.

1.3 Platinum Temperature Sensors

Platinum is characterised by its chemical inertness, time stability and high melting point. As a result, platinum is used in many temperature sensors. For operational sensors (e.g. PT100 or PT1000), the electrical resistance ratio $W_{100} = 1.385$ is prescribed by the standard IEC 60751 [3], and it is a key parameter for assessing platinum purity.

The relationship between platinum sensor resistance R and temperature ϑ is described by the Callendar-Van Dusen equations (1.4 and 1.3). The equation (1.4) shows resistance for temperature above 0°C and the equation (1.3) shows resistance for temperature below 0°C. [2].

$$\vartheta \in (-200; 0)°\mathrm{C}: \qquad R(\vartheta) = R_0 \cdot (1 + A\vartheta + B\vartheta^2 + C(\vartheta - 100) \cdot \vartheta^3) \tag{1.3}$$

$$\vartheta \in (0; 850)°\mathrm{C}: \qquad R(\vartheta) = R_0 \cdot (1 + A\vartheta + B\vartheta^2) \tag{1.4}$$

The coefficients in the Callendar-Van Dusen equations are defined by the IEC 60751 standard [3]. R_0 is the nominal resistance of the RTD at 0°C. For a PT100 RTD, the R_0 is 100 Ω. Coefficients A, B and C from the standard [3] are:

$$A = 3.9083 \cdot 10^{-3}\ \mathrm{K}^{-1},$$
$$B = -5.775 \cdot 10^{-7}\ \mathrm{K}^{-2},$$
$$C = -4.183 \cdot 10^{-3}\ \mathrm{K}^{-4}.$$

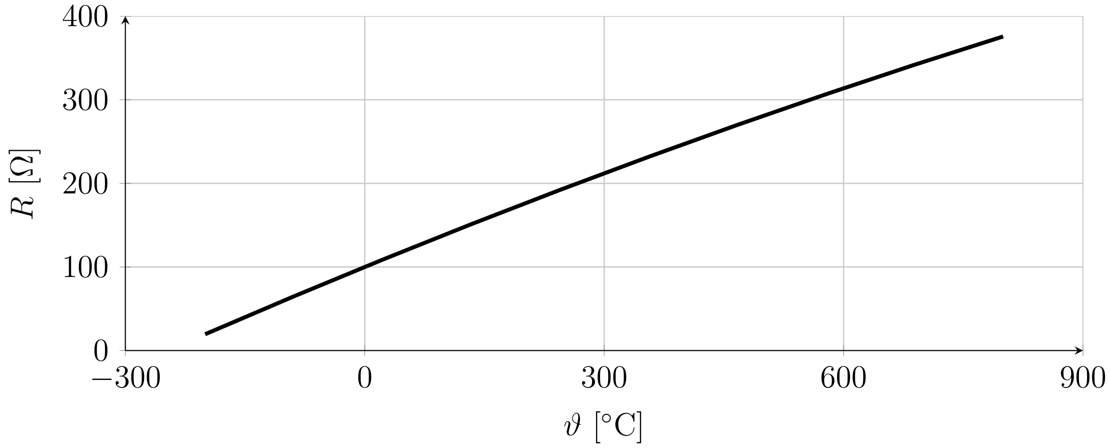


Fig. 1.1: The dependence of the PT100 resistance R on the temperature ϑ.

The graph in the Figure 1.1 shows the dependence of the resistance of the PT100 sensor on the temperature. Next Figure 1.2 shows the PT100 sensor non-linearity in the temperature range from 0°C to 100°C. The deviation in the middle of this range from an ideal characteristic is 0.145 %.

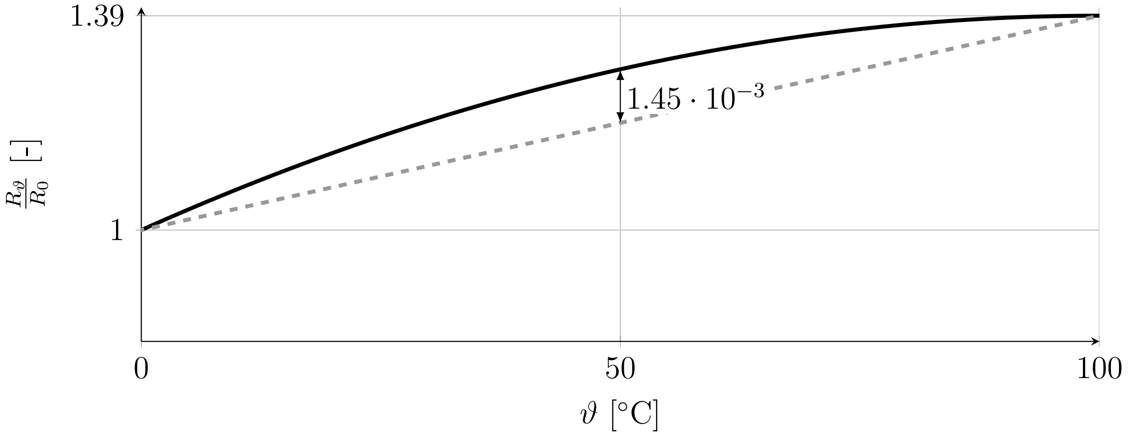


Fig. 1.2: Non-linearity of the PT100 in the temperature range 0°C to 100°C. [1]

1.4 RTD Tolerance standards

Two tolerance standards define a grade or a class for the platinum RTD accuracy. In North America is mainly used standard ASTM E1137 which represents two accuracy grades. On the other hand, the European standard is known as the standard IEC 60751, and it is used worldwide [2]. For this reason, the following text will refer exclusively to the IEC standard 60751. The standard IEC 60751 divides the sensors into several accuracy classes: AA, A, B and C. The Table 1.1 shows the classes' temperature range and their tolerances.

Tab. 1.1: Tolerance classes for platinum sensors by IEC 60751. [3]

Class	**Temperature Range** [°C]	**Tolerance Value** [°C]
AA	-50 to 250	$\pm(0.1 + 0.0017 \cdot \lvert\vartheta\rvert)$
A	-100 to 450	$\pm(0.15 + 0.002 \cdot \lvert\vartheta\rvert)$
B	-200 to 600	$\pm(0.3 + 0.005 \cdot \lvert\vartheta\rvert)$
C	-200 to 600	$\pm(0.6 + 0.01 \cdot \lvert\vartheta\rvert)$

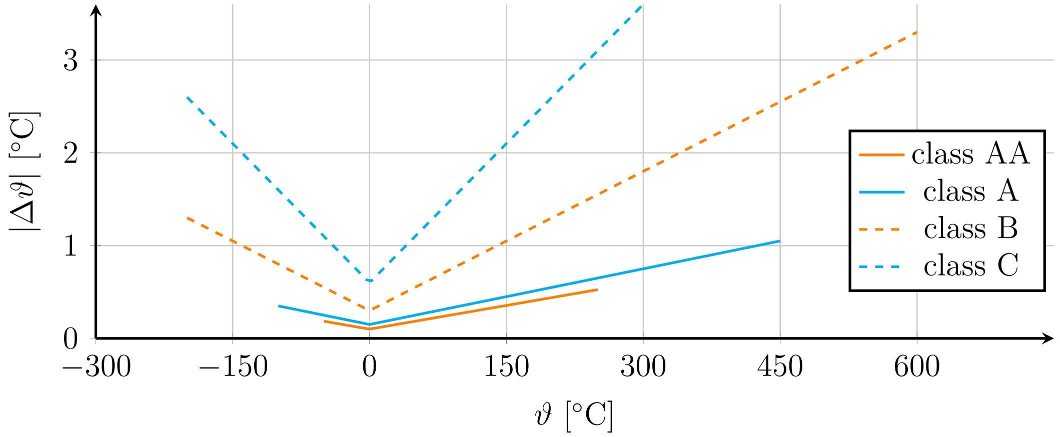


Fig. 1.3: The dependence of the tolerance value and measuring temperature by the standard IEC 60751. [3]

1.5 RTD measurement circuits

The first option is to connect RTD to a current source. A voltage dropout appears on the sensor. The voltage drop is measured. See the Figure 1.4. This voltage U_ϑ is measured directly by an A/D converter with subsequent digital processing, or the voltage drop could be measured with an analogue instrument (voltmeter). These measuring circuits are only suitable for linear sensors or sensors with little non-linearity. [1]

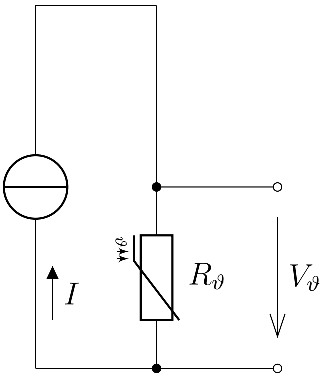


Fig. 1.4: Simple measuring circuit suitable for linear RTDs, redrawn from [1]

$$R_\vartheta = \frac{V_\vartheta}{I} \tag{1.5}$$

For non-linear sensors, it is necessary to linearise the sensor characteristic. Linearisation can be done either analogue (e.g. utilising the dedicated XTR103 circuit or utilising feedback in amplifiers) or digitally using a microcontroller with programmed inverse polynomials of the sensors or with a stored table of values where each voltage corresponds to a specific temperature. [1] Since the magnitude of the current I is constant and this value is known, Ohm's law applies to the measured voltage V_ϑ at the sensor.

1.5.1 2-Wire measurement

However, in a practical application, the sensor is not placed directly at the measurement terminals of the ADC. Therefore, the measurement is affected by the wiring resistance between the sensor and the measuring system, see the Figure 1.5. The resistance error cannot be separated from the RTD measurement. Two-wire RTDs yield the least accurate RTD measurements and are used when accuracy is not critical or when lead lengths are short. Two-wire RTDs are the least expensive RTD configuration [2]. The measured resistance is:

$$R_\mathrm{m} = R_\mathrm{w1} + R_\vartheta + R_\mathrm{w2}. \tag{1.6}$$

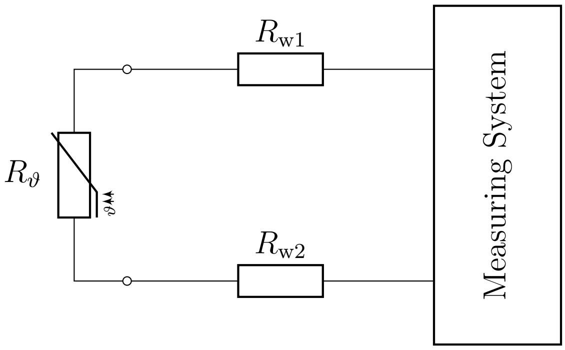


Fig. 1.5: Two-wire measurement circuit, redrawn from [1]

1.5.2 3-Wire measurement

The RTD is connected to two lead wires on one end and a single lead wire on the opposite end in the three-wire configuration. Using different circuit topologies and measurements, lead resistance effects can effectively be cancelled, reducing the error in three-wire RTD measurements. Compensation for lead wire resistance assumes that the lead resistances match.

Similar to the two-wire circuit, the total resistance of the measuring loop is given by:

$$R_\mathrm{m} = R_\mathrm{w1} + R_\vartheta + R_\mathrm{w2}. \tag{1.7}$$

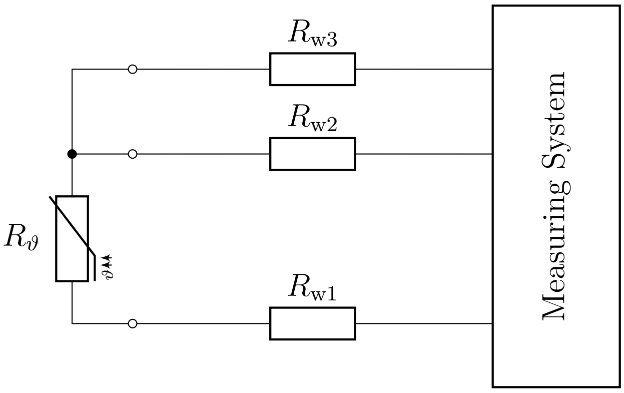


Fig. 1.6: Three-wire measurement circuit, redrawn from [1]

Assuming all resistances are equal $R_{\mathrm{w1}} = R_{\mathrm{w2}} = R_{\mathrm{w3}}$. We can simply measure the loop resistance given by:

$$R_{\mathrm{w23}} = R_{\mathrm{w2}} + R_{\mathrm{w3}}. \tag{1.8}$$

If we then subtract this resistance value R_{w23} from the measuring loop R_{m}, we get the resulting value of the measured resistance R_ϑ in our case PT100.

$$R_\vartheta = R_{\mathrm{m}} - R_{\mathrm{w23}} \tag{1.9}$$

1.5.3 4-Wire measurement

In the last four-wire configuration, two wires are connected to either end of RTD. In this configuration, the RTD resistance may be measured with superior accuracy. RTD excitation is driven through one wire on either end of RTD, while the next pair of wires is used for measuring the resistance. The RTD resistance is sensed without error from the wire reacting with the sensor excitation. Four-wire RTDs yield is the most accurate measurement but is the most expensive RTD configuration. [2]

1.5.4 Ratiometric measurement

The ratiometric measurement is typically used to measure RTD resistance value. The RTD resistance we can calculate from voltage drop on the RTD and from the current which flows thought the RTD.

$$R_\vartheta = \frac{V_\vartheta}{I_\vartheta} \tag{1.10}$$

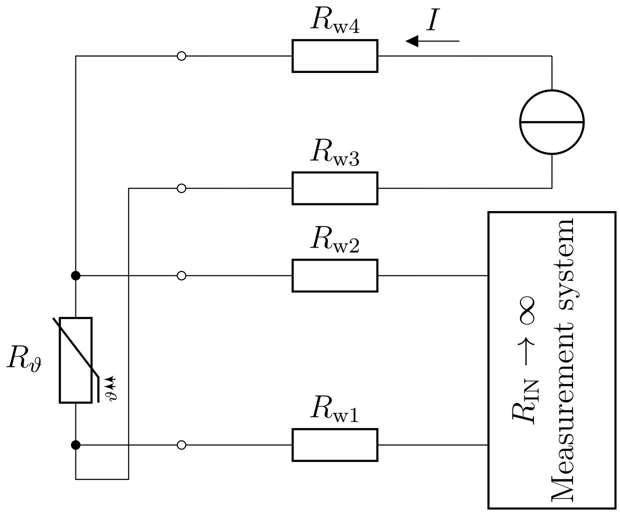


Fig. 1.7: Four-wire measurement circuit, redrawn from [1]

But the ADC can measure only voltage. Therefore we have to place another precise reference resistor. The ratio of the measured voltages on resistors corresponds to the ration of the resistances.

$$\text{ADC conversion result} = \frac{V_\vartheta}{V_{\mathrm{REF}}}(2^N - 1) = \frac{R_\vartheta \cdot I}{R_{\mathrm{REF}} \cdot I}(2^N - 1) = \frac{R_\vartheta}{R_{\mathrm{REF}}}(2^N - 1) \quad (1.11)$$

When we knew the value of the reference resistor we can calculate the resistance of the RTD:

$$R_\vartheta = \frac{\text{ADC conversion result}}{(2^N - 1)} \cdot R_{\mathrm{REF}}. \quad (1.12)$$

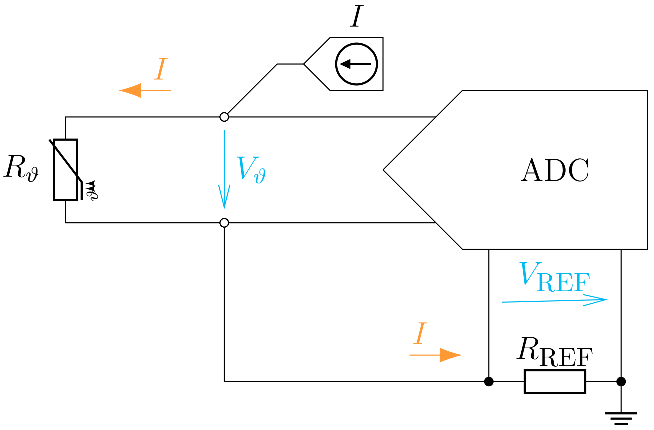


Fig. 1.8: Example of a ratiometric RTD measurement, redrawn from [2].

2 SPI

Serial peripheral interface (SPI) is one of the most widely used interfaces between the microcontroller and peripheral integrated circuits such as sensors, analog-to-digital or digital-to-analog converters, SD cards and others. This chapter briefly describes the main features and principles of the SPI communication.

The SPI is a high-speed synchronous full-duplex serial input and output (I/O) interface. The data from the master to the slave is synchronised by the rising and falling clock edge. Both master and slave can transmit data at the same time. The SPI interface can be either 3-wire or 4-wire. This chapter focuses only on the 4-wire SPI interface.

2.1 Interface

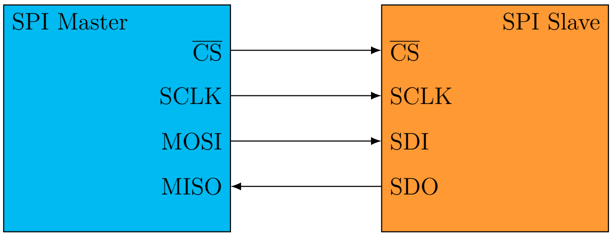


Fig. 2.1: SPI configuration with one master and one slave, redrawn from [4].

The four wires are used and their functions from the master side are:

- $\overline{\text{CS}}/\overline{\text{SS}}$ – Chip Select or Slave Select
- SCLK – SPI Clock
- MOSI – Master Output Slave Input
- MISO – Master Input Slave Output

The device that generates the clock signal is called the master. The clock signal synchronises the transmitted data between master and slave. In comparison with the I^{2}C bus interface, the SPI devices support much higher frequencies [4]. The manufacturer's specification determines the maximum transmission frequency [5].

The SPI specification says there can be only one master and one (or multiple) slave(s). The chip select master signal selects the device with which the communication will be established. During active transmission, the chip select is LOW, and it is pulled HIGH to disconnect the slave from the SPI bus and communication.

The vast majority of devices use $\overline{\texttt{CS}}$ LOW to activate data transfer [4]. It can be set to $\overline{\texttt{CS}}$ active HIGH for specific reasons, but all devices on the one SPI bus must be switched to this mode.

The `MOSI` and `MISO` pins are the data line. The `MOSI` transfers data from the master to the slave and `MISO` transfers data in the opposite direction.

2.2 Clock Polarity and Clock Phase

The master can select the clock polarity and also the clock phase. The clock polarity (CPOL) bit sets the polarity of the clock during the idle state. The idle state is defined as the period when $\overline{\texttt{CS}}$ is high and falling to low at the start of the transmission and when $\overline{\texttt{CS}}$ is low and rising to high at the end of the communication. The clock phase (CPHA) bit selects the clock phase. Depending on the CPHA bit, the rising or falling clock edge is used to sample and/or shift data. The master must select clock polarity and phase as required by the slave. Depending on the CPOL and CPHA selection, four SPI modes are available.

Tab. 2.1: SPI modes with CPOL and CPHA. [4]

SPI mode	CPOL	CPHA	Clock polarity in idle state	Clock Phase used to sample and/or shift the data
0	0	0	Logic Low	Data sampled on the rising edge and shifted out on the falling edge
1	0	1	Logic Low	Data sampled on the falling edge and shifted out on the rising edge
2	1	0	Logic High	Data sampled on the rising edge and shifted out on the falling edge
3	1	1	Logic High	Data sampled on the falling edge and shifted out on the rising edge

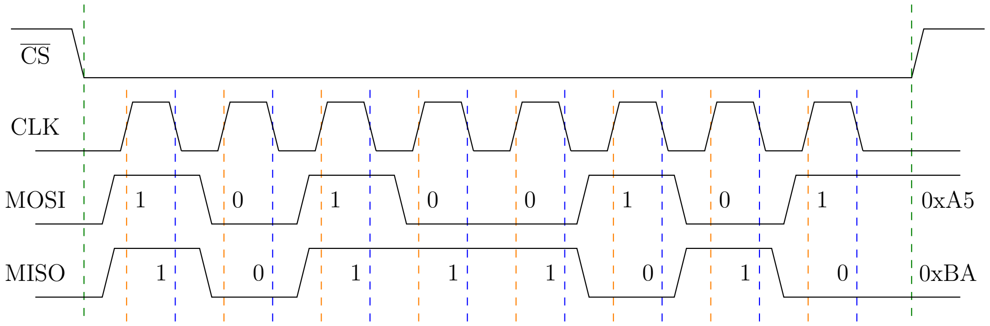


Fig. 2.2: SPI mode 0, CPOL = 0, CPHA = 0, CLK idle state = low. [4]

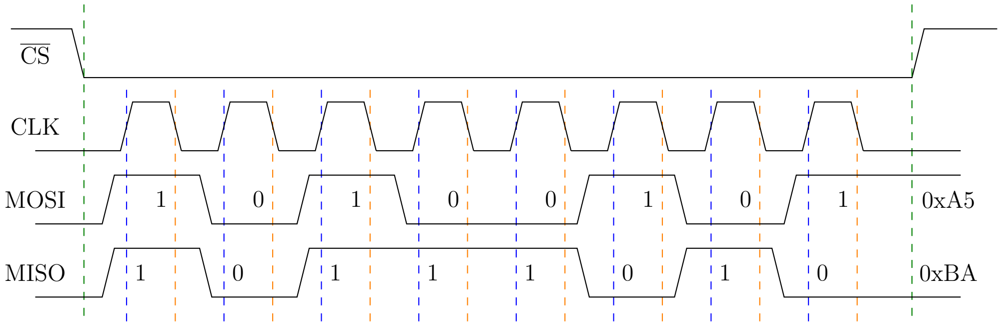


Fig. 2.3: SPI mode 1, CPOL = 0, CPHA = 1, CLK idle state = low. [4]

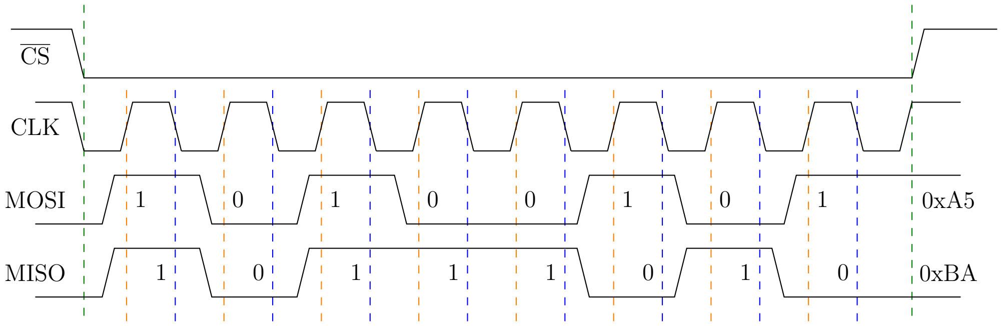


Fig. 2.4: SPI mode 2, CPOL = 1, CPHA = 0, CLK idle state = high. [4]

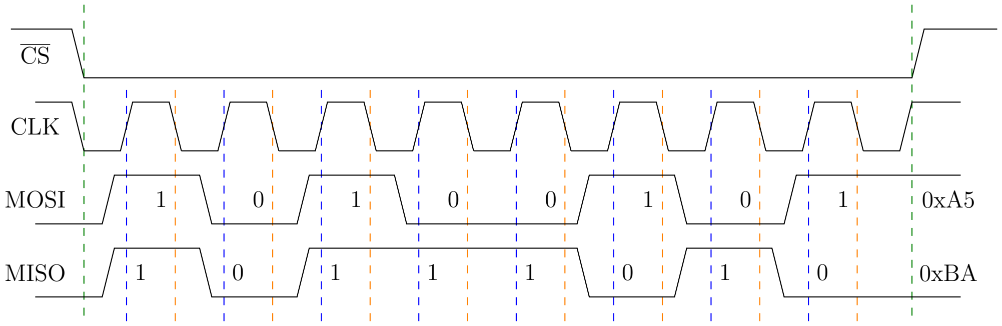


Fig. 2.5: SPI mode 3, CPOL = 1, CPHA = 1, CLK idle state = high. [4]

2.3 Multi slave configuration

Multiple slaves can be used with one main master. The slaves can be connected in the regular or Daisy-chain mode.

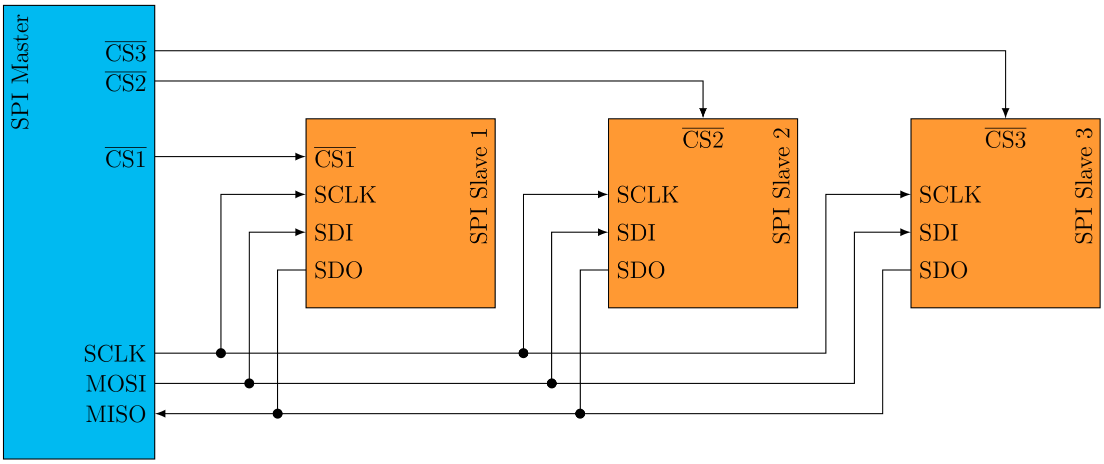


Fig. 2.6: Regular multi-slaves SPI configuration, redrawn from [4].

In the regular mode, an individual chip select line for each slave is required from the master. When the chip select is enabled (pulled low) by the master, the clock `SCLK` and data on the `MOSI`/`MISO` lines are available for the transferring data to and from the selected slave. If the multiple chip select signals are enabled, the data on the `MISO` line is corrupted, as there is no way for the master to identify which slave is transmitting the data. [4]

As can be seen in Figure 2.6, the number of chip select lines increases with the number of the slaves. This can soon take all the pins from the µC and the limit maximum number of the slaves. For this reason, there are many ways to avoid this problem. One frequently used example is the Daisy Chain method. In the Daisy-chain method, the slaves are configured that the $\overline{\texttt{CS}}$ signal selects all the slave devices. The `SDO` signal from the first device goes to the `SDI` of the second device. The data from the master is directly connected to the first slave, which provides data to the next slave. The Daisy chain is not supported by all SPI devices. Therefore, before implementing a slave, it is always necessary to check the manufacturer's documentation to verify that the slave supports the Daisy-chain method. Another way to reduce used µC pins is to use serial to parallel converters or special multiplexers. [4]

However, the used SPI device MAX31865 does not support the Daisy-chain method, and the number of slaves used in the design is not so high (only four), so we will not continue to focus on these line-reduction methods.

3 Bluetooth

This chapter describes Bluetooth technology. It explains the differences between Bluetooth Classic (BR/EDR) and Bluetooth Low Energy (BLE). Then it goes into more details about BLE. Main BLE parameters are described in subsection 3.2.1, limitations and advantages of BLE are described in subsection 3.2.2 and 3.2.3, respectively. Layer architecture of BLE is discussed in the last section 3.3.

3.1 Bluetooth Classic

Bluetooth is a short-range wireless communication standard used to exchange data over short distances between mobile devices. The most popular audio transfer feature is used during phone calls or playing multimedia. The BLE was initially conceived as a wireless alternative to the RS-232 standard for serial communication over wired lines. The first version of Bluetooth was standardized as IEEE 802.15.1 and released in 1994 by Ericsson. It was named after the Danish king Harald "Bluetooth". [6]

There are two types of Bluetooth. [7] One of them is known as Bluetooth Classic (BR/EDR), which is used, for example, in wireless speakers, headphones or car information systems. The other type is Bluetooth Low Energy (BLE). BLE was first introduced in Bluetooth version 4.0 in 2010. [6] Its main area of use is in applications where power consumption is critical and where only small amounts of data are transmitted. Therefore, it is mainly used in battery-powered devices, such as various sensors, detectors, IoT, smartwatches or smart bracelets.

These two types of Bluetooth standards are incompatible with each other, even though they share the same brand and even the document in which they are specified. Bluetooth Classic devices cannot communicate directly with BLE devices. For this reason, some devices, such as mobile phones or laptops, have both types (also referred to as dual-mode devices) implemented in them. Dual-mode allows them to communicate with both Bluetooth types. [8]

3.2 Bluetooth Low energy

Bluetooth Low Energy, known also as BLE, is a version of Bluetooth optimized for low power applications. Unlike the classic Bluetooth, the BLE frequency band is divided into 40 channels of 2 MHz width. Moreover, in BLE devices, data transfers occur in short bursts separated by standby periods, contributing significantly to

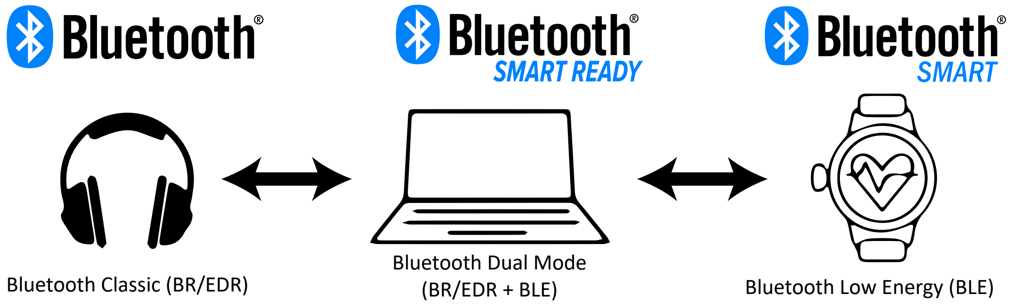

Bluetooth®
Bluetooth® SMART READY
Bluetooth® SMART
Bluetooth Classic (BR/EDR)
Bluetooth Dual Mode (BR/EDR + BLE)
Bluetooth Low Energy (BLE)

Fig. 3.1: Compatibility of the Bluetooth devices.

energy savings compared to always active Bluetooth devices. Most known fields of application of BLE include wearables, such as smartwatches and smartbands, Internet of Thing devices and a specific class of devices called beacons. [6]

3.2.1 Parameters

This section contains only the basic technical parameters related to Bluetooth Low Energy. More detailed information can be found in the BLE standard [7] issued by (Bluetooth SIG). The main parameters are:

- BLE operates in the unlicensed ISM frequency range from 2.402 to 2.480 GHz. [7]
- Frequency spectrum is divided into 40 channels where each channel has a width of 2 MHz. [7]
- The maximum transmission rate at the physical layer is 1 Mbps for BLE version 4.0 For BLE v5.0 and late, the maximum bit rate is up to 2 Mbps.
- A typical range is 10 to 30 meters (30 to 100 feet). The range varies significantly depending on the environment surrounding the communicating BLE devices and the transmit power setting (maximum 10 mW for BLE v4.0 and 100 mW for BLE v5.3). [7]
- Power consumption varies significantly depending on the implementation of the application itself, which uses BLE to transmit data. Consumption is also significantly affected by the settings of the BLE transmission parameters and the used hardware. The peak power consumption of the BLE chipset during transmission is typically under 15 mA. [6] However, the Nordic Semiconductor makes the SoC nRF52833 which can transmit data with radio consumption 6 mA at 0 dBm TX power [9].
- BLE communication security is optional, and it depends on the used hardware and developers whether to implement security in their application. If they choose, they have several levels of security which can be implemented (more

on this topic in the section 3.3 on the page 31).

- All encryption operations are secured using a 128 bit AES (Advanced Encryption Standard) encryption in CCM (Counter with CBC-MAC) mode.
- BLE is designed to transmit only low-bandwidth data transfer. When BLE is implemented in a high-bandwidth data application, the low power consumption is significantly compromised. The optimal power consumption is achieved if the radio usage time is as low as possible.
- All of the BLE versions are backwards compatible with each other. However, if the devices communicate with different versions, only the features supported by the older version can be used. For example, when Bluetooth 4.1 BLE-device communicates with device BLE version 5.0, it is not possible to use features from BLE version 5.0. [6]

3.2.2 Limitations

This subsection discusses the limitations and disadvantages of the BLE technology, mainly the data bandwidth and the radio range. These two parameters are often considered when wireless communication technology is selected for a new application.

Bandwidth

The bandwidth of the BLE is limited by the physical radio data rate, which means the rate of data transmission. This rate depends on the used Bluetooth version. For BLE version 4.2 and earlier, the rate is fixed at 1 Mbps. For Bluetooth v5.0 and late, the rate depends on the used transmission mode. The rate drops to either 500 or 125 kbps in the long-range mode. On the other hand, the high-speed mode rises up to 2 Mbps. [8]

The data rate at the application layer is much lower than the radio rate due to the following factors. Bluetooth specification defines a 150 µs gap between transmitting packets. During this gap, no valid data is transferred, and, as a result, the transfer rate at the application layer is reduced. [6]

Another factor is the packet header size. The larger the header, the lower the speed at the application layer compared to the rate at the physical layer, and Bluetooth is no exception. Sometimes, the central device will ask the peripheral to send data packets even though the peripheral has no useful data to send.

Packet loss is another possible factor affecting the transmission rate. If a packet is lost or damaged in transmission, the sender resends the lost packet. However, it takes time, and the transfer rate drops down.

Range

The BLE was designed for short-distance communication. Therefore, few factors limit its operation range. The main is used frequency. Bluetooth operates in the 2.4 GHz ISM radio spectrum, which is greatly affected by barriers all around us, such as metal objects, walls and water (especially human bodies).

Another critical factor affecting the range of BLE is the antenna used, its gain, orientation and location relative to the rest of the device. The material from which the antenna is made and the housing of the entire device can significantly affect the range of the device. In particular, if an internal antenna is used. [6]

3.2.3 Advantages

The main advantages of Bluetooth Low Energy are its low power consumption power, which is already apparent from its name. The specific power consumption is highly dependent on the transmission parameters' settings and the hardware used. [7]

Another significant advantage is its low purchase price. For most wireless protocols, it is common that access to the specification documents is charged (sometimes up to thousands of dollars per year). However, the BLE or BT BR/EDR specification is freely available in the specification [7] and anyone can download and implement it. In addition, the price of the BLE modules and chips themselves is lower compared to similar technologies. Another advantage is that BLE is embedded in most modern smartphones. It is probably BLE's most significant advantage over its competitors, such as ZigBee or Thread. [6]

3.3 Layer architecture of BLE

Layer architecture of the BLE is divided into three main blocks: *Application*, *Host* and *Controller*. The following subsections cover each layer. The layers are described from the lowest – Physical Layer to the highest abstraction layer – Application.

- Application
- Host
  - Generic Access Profile (GAP)
  - Generic Attribute Profile (GATT)
  - Attribute Protocol (ATT)
  - Security Manager (SM)
  - Logical Link Control and Adaptation Protocol (L2CAP)
  - Host Controller Interface (HCI)
- Controller
  - Host Controller Interface (HCI)
  - Link Layer (LL)
  - Physical Layer (PHY)

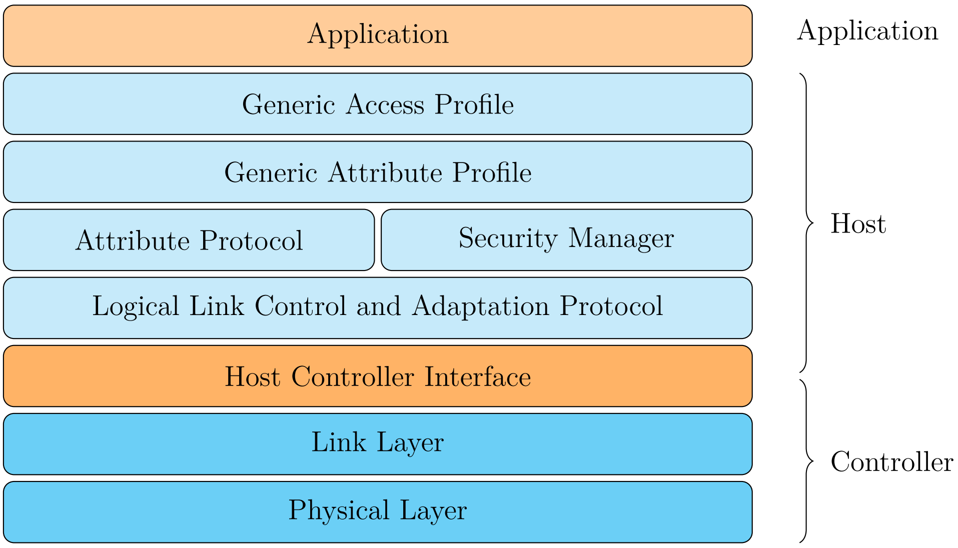


Fig. 3.2: Bluetooth Low Energy architecture, redrawn from [6].

3.3.1 Physical Layer

The physical layer is responsible for transmitting and receiving packets of information on the physical channel. The PHY block transforms the data stream to and from the physical channel and the baseband into required formats.

The BLE operates in an unlicensed ISM band in the frequency range from 2402 MHz to 2480 MHz. This range is divided into 40 radio channels, where three channels are used for scanning and advertising and the remaining 37 for sending data. The centre frequencies of the adjacent channels are always 2 MHz apart [7]. The frequency spectrum and radio channels can be seen in the Figure 3.3.

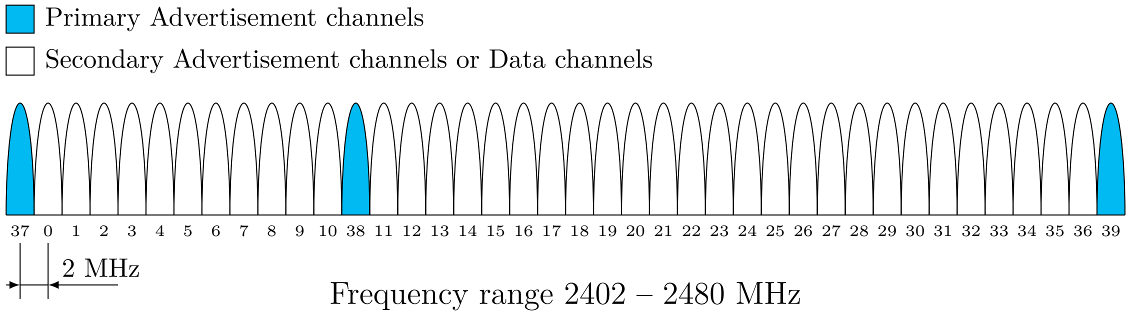


Fig. 3.3: Frequency spectrum and RF channels in BLE, redrawn from [6].

BLE uses Frequency Hopping Spread Spectrum (FHSS), which allows both the two communicating devices to randomly change the carrier frequency on which the data is sent for a period of time. This method significantly improves transmission reliability and allows the devices to avoid frequency channels used for other communications or are jammed.

The transmitting power of the radio can be set in a range [7]:

- Minimum output power 0.01 mW (−20 dBm)
- Maximum output power 100 mW (+20 dBm)

3.3.2 Link Layer

The link layer is directly connected to the physical layer and provides a degree of abstraction to the higher layers. It also provides the ability to interact with the radio through the HCI layer. The link layer is primarily responsible for calculating the cyclic redundant (CRC), random number generation, encryption, and state management in the device. A device can always be in one of these states [7]:

- Standby state
- Advertising state
- Scanning state
- Initiating state
- Connection state
- Synchronisation state
- Isochronous broadcasting state

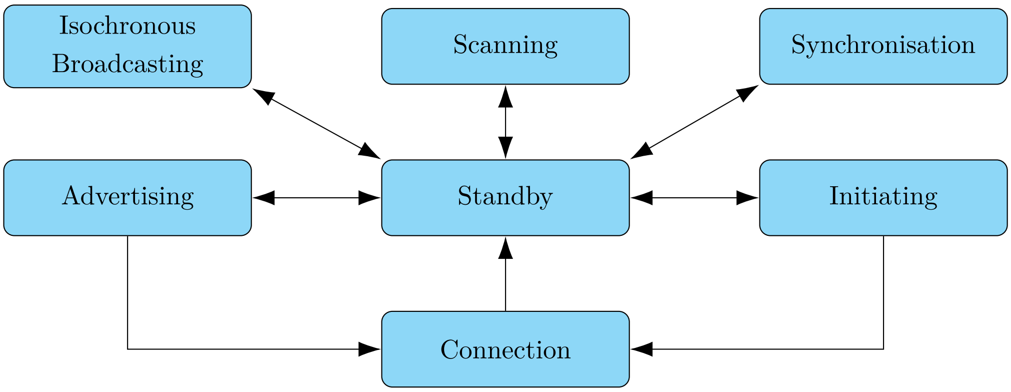


Fig. 3.4: State diagram of the Link Layer state machine, redrawn from [7].

The link layer in the Standby state does not transmit or receive any packets.

The link layer in the Advertising state transmits the physical advertising channel. Therefore, the device in the Advertising state is known as an advertiser.

The Scanning state is used to listen to advertising physical channel packets from advertising devices. A device in the Scanning state is known as a scanner.

The device in the Initiating state listens for advertising physical channel packets and responds to initiate a connection with another device. A device in the Initiating state is known as an initiator.

The devices in the Connection state are connected, and data is exchanged regularly.

The link layer in the Synchronisation state listens for periodic physical channel packets forming a specific periodic advertising train, coming from a specified device that is transmitting regular advertising.

The Link layer in the Isochronous Broadcasting State will transmit isochronous data packets on an isochronous physical channel.

3.3.3 Host Controller Interface

The Host Controller Interface layer is a standard protocol defined by BLE specification that allows the host to communicate with the controller. Host and controller could exist in separate chipsets, and the HCI ensures their compatibility. The device developer can then choose different manufacturers for the control and host parts, and he can be sure that HCI will make interoperability possible. [8]

The HCI is implemented through a physical interface if the host and controller parts are in physically separate chips. Officially supported interfaces are UART

(Universal Asynchronous Receiver-Transmitter), USB (Universal Serial Bus) and SDIO (Secure Digital Input Output). The HCI layer is implemented using a logical interface if the control and host parts are in the same chip. The main task of the HCI layer is to transfer commands from the host (ATT commands) to the control part and also to transfer events in the opposite direction. [6]

3.3.4 Logical Link Control and Adaptation Protocol

Logical Link Control and Adaptation Protocol (L2CAP) is a kind of protocol multiplex. It is the same protocol used in Bluetooth Classic. It takes data from the upper layers and places them into standard Bluetooth packets, which it then forwards to the lower layers. In the case of BLE, it primarily receives data from the Attribute Protocol layer and the Security Manager layer. [6]

It also handles packet segmentation and reassembly. It splits large packets from higher layers into pieces that fit into the space reserved for valuable data in Bluetooth packets. The maximum size of payload data depends on the BLE version. The maximum payload data size for versions 4.0 and 4.1 equals 27 B. From version 4.2, the maximum payload data size is 247 B. On the receiver side, this layer takes several Bluetooth packets and reconstructs the higher layer packets from them. Encapsulation of ATT layer data into BLE packet is shown in the following Figure 3.5.

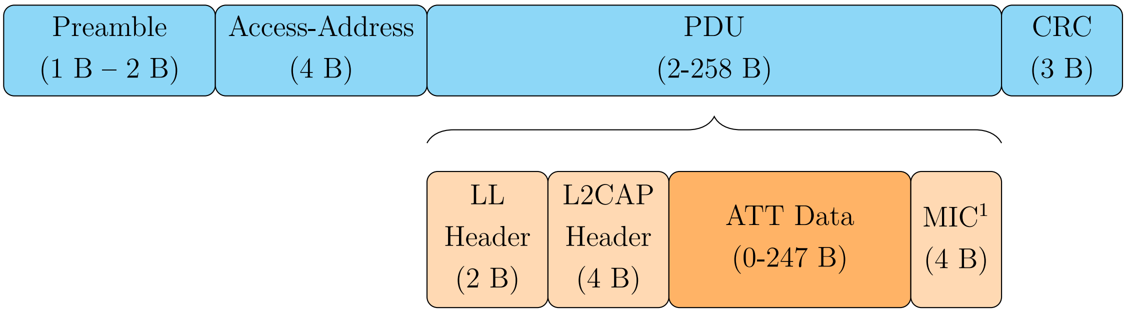


Fig. 3.5: Packet format for uncoded BLE data packets, redrawn from [7].

3.3.5 Security Manager

The Security Manager Protocol handles security in BLE, which is the peer-to-peer protocol used to generate encryption keys and identity keys. The security manager block is used only in BLE systems. Similar security functionality in BR/EDR system is contained in the Link Manager block in the controller [7]. The security layer uses

[1] Optional [7]

several protocols and algorithms needed to ensure security. The main functions of this layer are:

- the use of whitelists, which allow filtering devices during scanning, advertising and connection establishment,
- creating and storing shared secret keys between devices,
- providing authentication,
- encryption of data exchanged between devices (128-bit AES cypher),
- ensuring data integrity using digital data signatures.

3.3.6 Attribute Protocol

Attribute Protocol (ATT) implements the peer-to-peer protocol between an attribute server and an attribute client. It defines how a server exposes its data to a client and how this data is structured. There are two roles of the ATT: *Server* and *Client*. [7]

In our project of the wireless thermometer, this device would be a server in the understanding of the ATT protocol. A connected device such as a smartphone or PC would be the client that connects to the wireless thermometer and requests the values of the sensor temperatures and others device attributes.

Each stored value, typically a few bytes, is known as an attribute. The ATT allows each attribute to be self-identified using UUIDs to identify the data type. These UUIDs can be well-known 16-bit assigned numbers defined in the Assigned Numbers document [10] and associated specifications. Or a manufacturer can assign their 128-bit UUID. An example of the UUID can be:

```
0000ABCD-5468-616E-6B73-4D6F6D446164
```

The attribute permissions indicate whether the attribute can be written to, read from, notified, or indicated. These permissions are not directly discoverable by ATT but rather through higher layers (GATT or up to the application layer). Up to six possible operations can be performed:

- Commands
  - They are sent by the client to the server and do not require a response.
- Requests
  - Are sent by the client to the server and require a response.
- Response
  - Is sent by the server to the client after the request is completed.

- Notifications
  - Is sent by the server to the client to notify the client of a change of a specific characteristic value and does not require confirmation from the client.
- Indication
  - Is sent by the server to the client to notify the client of a change of a specific characteristic value and confirmation is required from the client.
- Confirmation
  - Is sent by the client to the server upon successful receipt of an indication.

3.3.7 Generic Attribute Protocol

The Generic Attribute Protocol (GATT) defines the communication form after pairing devices. The structure of the GATT is shown in the Figure 3.6. The GATT comes with the following terms:

- Profiles
- Services
- Characteristics

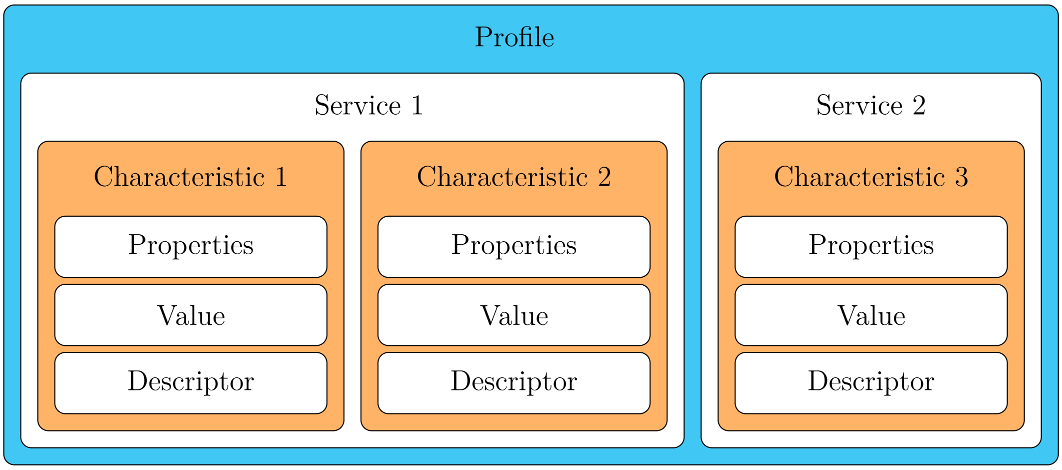


Fig. 3.6: GATT-Based Profile hierarchy, redrawn from [7].

The BLE profile collects the services. Bluetooth SIG could predefine it, or the developer could define his own profile with his own services. One profile contains not only one service but often more services. [8] For example, the health thermometer profile has two services: the first is called Health Thermometer Service and transfers the measured temperature data, and the second – Device Information Service contains state information about the device. [11]. The BLE service is used to sort the BLE profile data logically.

The BLE characteristic is similar to an attribute from the ATT protocol. The GATT layer comes with the same concept of client and server as the ATT layer. But the difference is that one device can change its role depending on the direction of the current communication. At one time, a device may play the role of a client requesting information from a server, and at another time, it may be the other way around. The Figure 3.6 shows how the characteristics and services are arranged in a profile.

3.3.8 Generic Access Profile

Broadcaster role

A device in the Broadcaster role is a device that sends advertising events or periodic advertising events and may also send Broadcast Isochronous Stream (BIS) events. The device operating in the Broadcaster role is called Broadcaster and shall have a transmitter and may have a receiver.

Observer role

A device operating in the Observer role is a device that receives advertising events or BIS events from another device which is in the Broadcaster role. A device running in the Observer role is called Observer and shall have a receiver and may have a transmitter.

Peripheral role

This role corresponds to the role of a slave on the link layer. The device performing the peripheral role sends out advertising events, thereby exposing itself to the central device and allowing the connection between them to be established. The peripheral role does not require powerful HW to run, making peripheral devices low-cost.

Central role

This role corresponds to the role of the master on the link layer. The device with the central role can establish several connections with peripheral devices and is always the initiator of the connection. A device usually performs this role with more computing power (e.g. mobile phone, personal computer). Establishing and maintaining several connections simultaneously requires a more powerful CPU and more memory.

4 Hardware design

The functionality of each component is described in this chapter. At the beginning of the chapter, the focus is on the power supply part (battery and LDO regulator). Then the main measuring circuit of the MAX31865 is described. After that, the description focuses on SPI communication and bus buffers. Finally, this chapter concludes with a discussion of PCB design, component placement and PCB balancing.

4.1 Power supply

The power supply provides the energy for the device's correct operation, and therefore, it is essential for the HW design. The µC nRF52833 can operate in voltage range from 1.7 to 5.5 V [9]. The temperature measuring IC MAX31865 operates in the range from 3.0 to 3.6 V with the typical voltage level of 3.3 V [5]. Therefore, the main board power supply voltage is selected at 3.3 V.

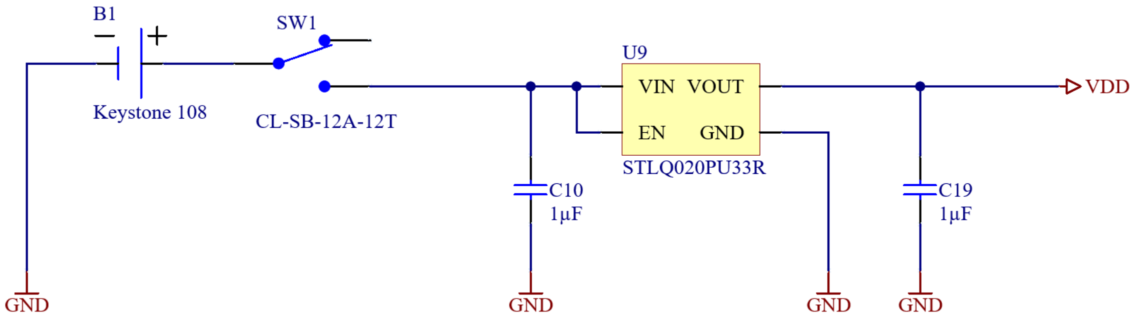


Fig. 4.1: Power supply.

4.1.1 Battery

For powering the board, the battery was selected. The rotor cup diameter is 68 mm which is enough to place the standard AA[1] size battery. The board will be placed on the top of the rotor cup and rotated by 7100 RPM. To reduce centrifugal forces, it is necessary to place the smallest possible battery with a voltage level above 3.3 V.

The battery SL-750/S from the TADIRAN manufacturer is chosen. It is a 1/2 AA size battery with a voltage level of 3.6 V. It is enough to power peripherals at the voltage level of 3.3 V. The battery voltage of 3.6 V indicates the battery composition from Lithium. By the datasheet [13], the nominal capacity of the SL-750 is 1100 mAh, and the temperature range is from −55°C to +85°C. We do not assume

[1] The IEC 60086 system calls the AA size as R6. [12]

a temperature greater than +85°C on the rotor surface. If it is necessary to use a battery for a higher temperature (above +85°C), the alternative could be SL-550 with an extended temperature range up to +130°C [14].

The battery holder type 108 [15] was chosen from the Keystone manufacturer because there is an option to use a cover 108C [16] that securely holds the battery against falling away. Minus pin of the battery is connected to the ground of the PCB design and the plus pin of the battery is connected to the main disconnector SW1 which is used for turning on and off the power supply.

4.1.2 LDO regulator STLQ020PU33R

The low-dropout voltage regulator STLQ020PU33R reduces the battery voltage to the operating voltage level 3.3 V. The maximum input voltage rating is 5.5 V, and the maximum output current is up to 200 mA, which is enough for the designed application. This regulator was chosen for its good stability and accuracy. The regulator itself regulates the output current and power, and it can operate from −40°C to +125°C. The STLQ020 embeds a constant-current limit circuit, which acts in case of overloads or short-circuits on the output, clamping the load current to save value (typ. 380 mA). [17]

Tab. 4.1: Electrical and thermal characteristics of LDO 3.3 V STLQ020. [17]

Symbol	Parameter	MIN	TYP	MAX	Unit
V_{IN}	Operating input voltage range	2		5.5	V
V_{OUT}	Output voltage	3.2	3.3	3.4	V
V_{DROP}	Dropout voltage ($I_{\mathrm{OUT}} = 20$ mA)		15		mV
I_{q}	Shutdown current $V_{\mathrm{EN}} = 0$ V		5	50	µA
I_{SC}	Short-circuit current $V_{\mathrm{OUT}} = 0$ V		380		mA
V_{EN}	Enable input logic low			0.4	V
	Enable input logic high	1.2			
$T_{\mathrm{J-OP}}$	Operating junction temperature	−40		+125	°C

In addition to the current limitation, the STLQ020 also embeds the internal thermal feedback loop, which disables the output voltage if the die temperature reaches approximately 160°C. This feature protects the device from excessive temperature that could permanently damage the LDO. The output voltage appears after the temperature drops below 140°C due to hysteresis. [17]

By the datasheet [17], it is recommended to place input and output external capacitors to assure the control loop stability. These capacitors must be selected to meet the requirements of minimum capacitance and equivalent series resistance. The capacitors C10 and C19 are placed as close as possible to the input and output pins. Both capacitors have capacitance 1 µF and are X7R type.

From this point, the main supply voltage for the whole board is designated as VDD.

4.2 MAX31865

The main measuring circuit is MAX31865 from the production of Maxim Integrated company. The MAX31865 is an easy-to-use resistance-to-digital converter optimised for measuring the resistance of platinum resistance temperature detectors (RTDs). The external reference resistor sets the sensitivity of precision delta-sigma ADC. The MAX31865 converts the ratio of the RTD resistance to the reference resistance into the digital form. The SPI peripheral bus is used for interfacing MAX31865 with the master µC.

The MAX31865's inputs are protected against overvoltage faults up to ±45 V. Programmable detection of the RTD and open and short-circuit conditions are included. The recommended DC operation conditions by datasheet [5] are shown in the Table 4.2.

Tab. 4.2: The recommended DC operation conditions and characteristics. [5]

Symbol	**Parameter**	**MIN**	**TYP**	**MAX**	**Unit**
V_{DD}	Operation voltage range	3.0	3.3	3.6	V
I_{DD}	BIAS off, ADC off		1.5	3	mA
	BIAS on, active conversion		2.0	3.5	mA
V_{BIAS}	Measuring BIAS voltage	1.95	2.0	2.06	V
I_{OUT}	Measuring BIAS current	0.2		5.75	mA
R_{REF}	Reference resistor	350		10k	Ω
t_{BIAS}	Bias voltage startup time			10	ms
t_{CONV}	single conversion time (50Hz filter)		62.5	66	ms
V_{IL}	Input digital logic 0	-0.3		$0.3 \cdot V_{\mathrm{DD}}$	V
V_{IH}	Input digital logic 1	$0.7 \cdot V_{\mathrm{DD}}$		$V_{\mathrm{DD}} + 0.3$	V

4.2.1 Power supply

Battery applications require the lowest possible power consumption for a long life because the amount of energy stored in the battery is not infinite. The current consumption of the MAX31865 during measuring resistance and ADC conversion is typically 2.0 mA and could be up to 5.75 mA.

The requirement for the temperature measurement is a sampling rate of 1 Hz or less. For this reason, it does not make sense that the MAX31865 is permanently turned on and wastes the power all the time. It will be better to use "deep sleep mode" and only for measurement requests, wake up the MAX31865, measure temperature and then put it back to sleep. But there is no functionality similar to the deep sleep mode. [5] It is just possible to turn off the BIAS voltage to reduce power consumption, but the MAX31865 still takes 2 mA. For this reason, each MAX31865 is powered by an integrated load switch MIC94300 which enables voltage on its output when the enable pin is in logic HIGH. The logic LOW disables the signal on the output. If the EN input pin is connected to the µC, The MAX31865 can be controlled remotely from the control program, and IC can be turned off during idle time and save the battery power.

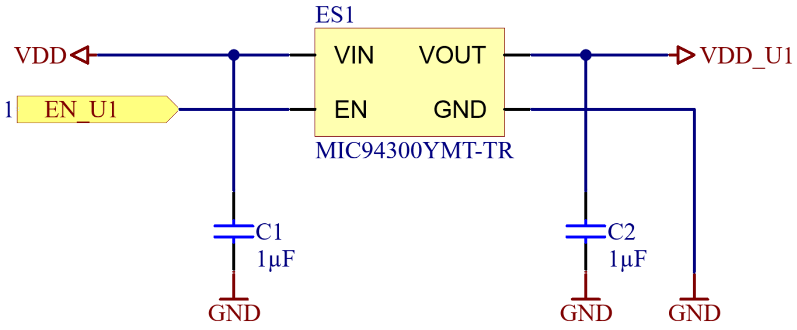


Fig. 4.2: Load switch ES for the powering MAX31865.

The electrical switch ES1 voltage input pin VIN is connected to the VDD voltage level. Input pin EN is connected to the µC and the net is labeled EN_U1 which enables powering for U1 (MAX31865). The output pin VOUT is connected to the peripherals that are to be remotely controlled. For better clarity the net for U1 device is labeled VDD_U1. And finally, it is clear to connect pin GND to the main ground of the board.

The manufacturer requires a minimum capacity of 470 nF for input and output capacitors. But for optimal ripple rejection performance, a 1 µF is recommended. The input and output capacitors C1 and C2 have X7R ceramic dielectric with low-ESR. And should be placed as close as possible to the ES1. [18]

4.2.2 LED indicator

In some cases, it is necessary to know which sensor is used to measure and which is not. For this purpose, LED indicators are used. Each MAX31865 has its own LED, which is powered by the same power supply as the MAX31865. The LED is placed near the corresponding IC. The soldering bridge `SB1` allows the user to use the `LED1` or leave it off to save power consumption.

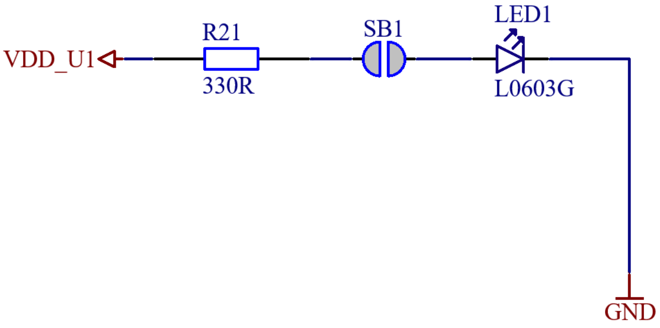


Fig. 4.3: Circuit diagram of the status LED.

4.2.3 SPI communication

The MAX31865 is the peripheral IC which means it needs a controlling microprocessor to operate. The MAX31865 uses the SPI bus as a communication interface for communication with the master µC. (More about SPI bus implementation and controlling the MAX31865 is in the section 5.1 on the page number 53.)

The designed wireless thermometer is a 4-channel thermometer, which means that four sensors and four MAX31865 ICs are used. Each is connected to the same SPI bus. The SPI assumes that all slaves are turned on and can participate in the communication in the normal condition. However, if one (or more) of the slaves is turned off, it still draws current from the SPI bus during the communication with other slaves and therefore, the turned slave loads the microcontroller SPI pins. From an electronic point of view, each SPI line inside the MAX31865 is connected via a diode to the `VCC` and `GND`, which means that if a voltage appears on an SPI line during communication with other slaves, current will flow to the MAX31865 `VDD` and power it. This condition is not allowed and must be resolved. For this reason, a buffer SN74LVC125A is placed between each IC MAX31865 and µC. The buffer ensures the impedance separation between each IC and µC. More about buffers is in the section 4.3 on the page 44.

4.2.4 Measuring RTD

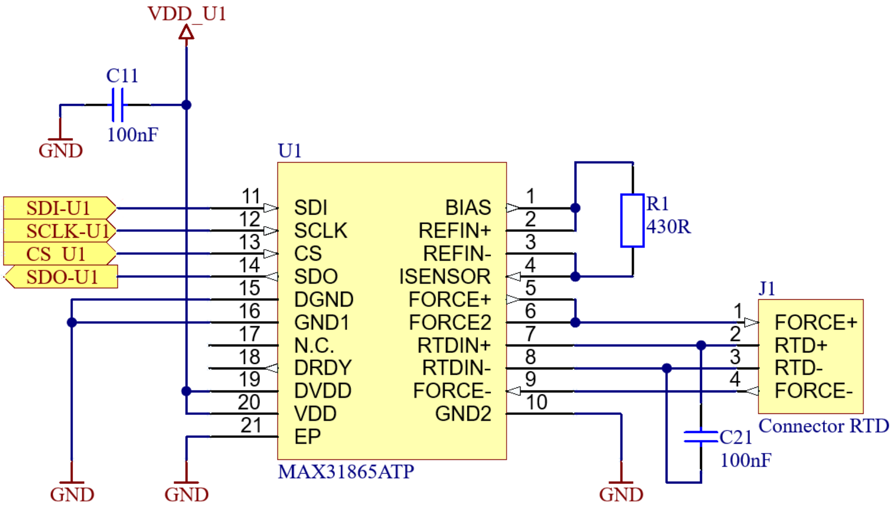
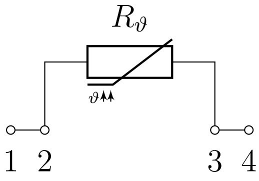
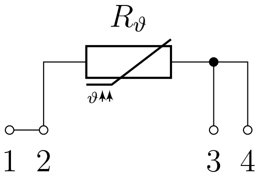
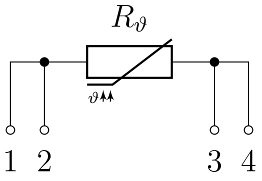


Fig. 4.4: Circuit diagram of the MAX31865.

The Figure 4.4 shows the typical application circuit diagram that allows measuring the PT100 RTD by 2, 3 or 4-wire sensor connection. The connector `J1` is used to connect the RTD sensor. The Table 4.3 shows the possible wire configuration of the RTD measurements. The `R1` is the reference resistor which should be as accurate as possible. The tolerance should be ±1 % or better. By the datasheet [5], the value of the reference resistor varies depending on the RTD sensor value. If the PT1000 sensor is connected, the value of the reference resistor should be 4300 Ω. The default value of 430 Ω is sufficient for PT100.

Tab. 4.3: RTD wire configuration (terminal `J1` to `J4`).

2-wire	**3-wire**	**4-wire**
		

The capacitors `C11` and `C21` are the decoupling and filter capacitors placed according to the manufacturer recommendations – as close as possible to the MAX31865 IC. The `VDD_U1` is the supply voltage from the electronic load switch `ES1` as was described in the subsection 4.2.1 on the page 41.

4.3 Quadruple Bus Buffer SN74LVC125A

This quadruple bus buffer gate is designed for 1.65 V to 3.6 V operation. It offers 4 independent channels, and each of them could be in 3 states logic high, logic low, and floating. The last floating state is when the corresponding port is disabled when output-enable ($\overline{\mathtt{OE}}$) input pin is high. The floating output is also when the buffer is switched off.

The bus buffers are used for each MAX31865, and they are placed near the corresponding IC. Each input ports (`SDI`, `SCLK` and $\overline{\mathtt{CS}}$) are still enabled by grounded corresponding $\overline{\mathtt{OE}}$ but output `SDO` is disabled when the SPI communication is not established with the MAX31865. When the master sets the communication with the MAX31865, the corresponding $\overline{\mathtt{CS}}$ is pulled down, and it enables the output `SDO`. If this was not ensured, a short circuit could occur if one buffer gives a logic high to `SDO` and the other gives a logic low to `SDO` at the same time.

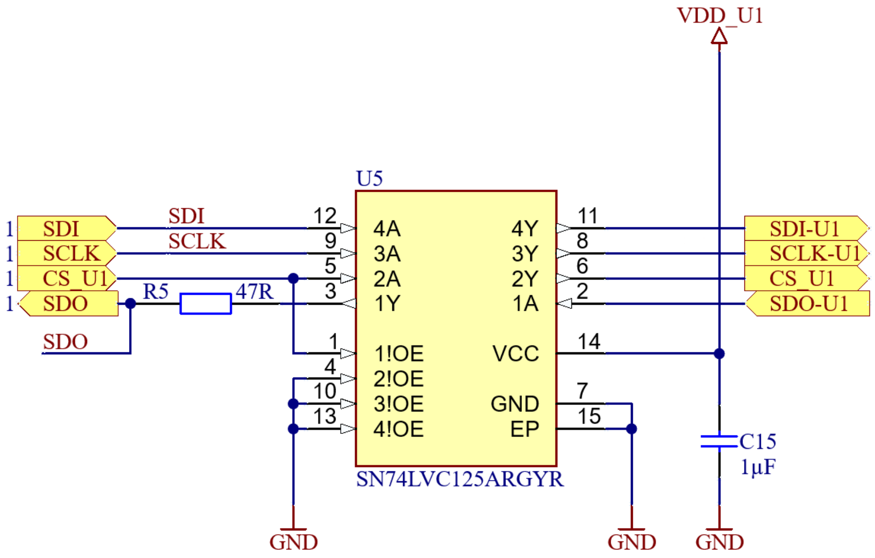


Fig. 4.5: Circuit diagram of the bus buffer.

4.4 nRF52833

The nRF52833 is an ultra-low-power System on a Chip (SoC) manufactured by Nordic Semiconductor. It is qualified for operations at the temperature range from −40°C to +105°C. Its feature set fulfils the requirements of professional lighting, advanced wearables, and higher value IoT applications. It supports standard IEEE 802.15.4, Bluetooth LE, Bluetooth mesh, Thread, Zigbee and proprietary 2.4 GHz protocols. [9]

Tab. 4.4: Electrical and thermal characteristics of the buffer SN74LVC125A. [19]

Symbol	Parameter	MIN	TYP	MAX	Unit
V_{CC}	Operating voltage range	1.65		3.6	V
V_{IH}	High-level input voltage $V_{\mathrm{CC}} > 2.7$ V	2.0		5.5	V
V_{IL}	Low-level input voltage $V_{\mathrm{CC}} > 2.7$ V			0.8	V
V_{O}	Output voltage $V_{\mathrm{CC}} > 2.7$ V	0		V_{CC}	V
T_{A}	Operating ambient temperature	−40		+125	°C

The nRF52833 is based on the ARM Cortex-M4 core with a floating-point unit (FPU). It has 512 KB flash and 128 KB RAM. The key radio features are BLE v5.3, direction finding, long-range and Bluetooth mesh, the output TX power is up to +8 dBm and sensitivity −95 dBm in 1 Mbps speed.

Peripheral features are EasyDMA, Full-speed 12 Mbps USB 2.0, high-speed SPI (up to 32 MHz), UART, I^{2}C, PWM, and 12-bit ADC. The 1.7 to 5.5 V supply voltage range enables powering the device from batteries or over USB.

The nrRF52833 is manufactured in three packages of different sizes, and three-part manufacturers also use it for manufacturing Bluetooth modules. One of these Bluetooth module could be a BL653µ from Laird Connectivity. [20] Main advantages of this BT module are its compact size and integrated antenna with mandatory components in one package. Furthermore, the easy-to-use module BL653µ is used in our design of a wireless thermometer. Therefore, the electrical and thermal characteristics in Table 4.5 are related to the BL653µ module, and also the pin assignments in Tables 4.6 and 4.7 corresponds with BL653µ pins.

Tab. 4.5: Electrical and thermal characteristics of the nRF52833. [21]

Symbol	Parameter	MIN	TYP	MAX	Unit
V_{DD}	Operating voltage range	1.7	3.0	3.6	V
I_{ON}	System ON, wake on any event		1.8		µA
I_{RADIO}	Radio transmitting 0 dBm output power		6.0		mA
T_{A}	Operating ambient temperature	-40		105	°C

4.4.1 Power supply

The microcontroller is powered directly from the main power supply voltage `VDD` on the voltage level of 3.3 V. The typical voltage level is 3.0 V, but the manufacturer allows to use of nRF52833 in voltage level up to 3.6 V. For powering, for example, from USB, the high voltage `VDDH` pin has to be used. The nRF52833 also contains an LDO regulator, which is used in high-voltage mode. The designated wireless thermometer is powered by 3.6 V battery. Therefore, the high-voltage mode will not be used.

4.4.2 SPI bus

The nRF52833 allows to use up to 4 SPI master instances: 3 with normal bit rate 8 Mbps, and the last SPI master 3 (SPIM3) instance enables bit rate up to 32 Mbps. There are no obligatory pins which mean each free GPIO pin can be used[2] for SPI bus. The assignment of the used pins for the SPI bus is in the Table 4.6. Each SPI line has its own serial resistor to reduce ringing caused by the output driver's impedance. The serial resistors should be placed as close as possible to the output pins.

Tab. 4.6: Pin assignments – SPI bus of nRF52833.

Description	**nRF52833 pin**	**BL653µ pin**	**Note**
SCLK	P0.27	40	SPI bus clock signal
MOSI	P0.29	39	Master output slave input (or SDI)
MISO	P0.30	38	Master input slave output (or SDO)
CS_U1	P0.23	8	Chip select U1
CS_U2	P0.24	4	Chip select U2
CS_U3	P0.17	28	Chip select U3
CS_U4	P0.31	12	Chip select U4

4.4.3 Enable MAX31865 powering signals

The reasons for turning MAX31865 off and on are described in the subsection 4.2.1. Each MAX31865 has its own load switch `ES` which can be turned on or off depending on the state on its `EN` pin. The following Table 4.7 shows the pin assignment for each signal `EN_U1`–`EN_U4`. The load switch manufacturer says the `EN` pin cannot be left floating. [18] For this reason, the internal nRF52833 pull-down resistors will be used for pulling output pins down.

[2] When the SPIM3 is used, the only high speed GPIO pins has to be used. [21]

Tab. 4.7: Pin assignments – Enable signals.

Description	nRF52833 pin	BL653µ pin	Note
EN_U1	P0.03	10	Enable signal U1
EN_U2	P0.12	29	Enable signal U2
EN_U3	P0.14	27	Enable signal U3
EN_U4	P0.08	14	Enable signal U4

4.4.4 Reset button

The reset button is connected to the pin P0.18 of the nRF52833. [21] When the reset pin is high, the program run is enabled. The logic low disables the program. The `nRESET` net is connected to the nRF52833 pin P0.18 (pin 6 of BL653µ). This net is pulled up by the resistor `R31`. The button `SW2` switches the `nRESET` net to the ground and holds the µC in reset until the button `SW2` is released. The capacitor `C20` disables µC in the first several milliseconds after turning on until the voltage `VDD` is levelled.

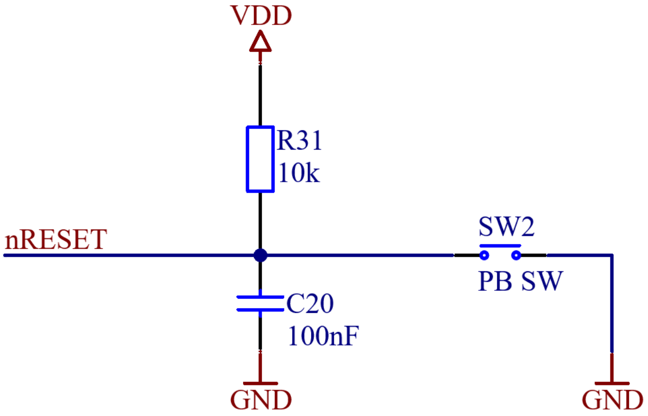


Fig. 4.6: Reset button.

4.4.5 Flash and Debug connector

The flash and debug connector is a standard ARM 10-pin SWD connector. The connector `P1` is the smallest possible SWD connector, and it allows the flashing and verifying of the program in the µC. In addition, it allows the debugging of a program in the µC.

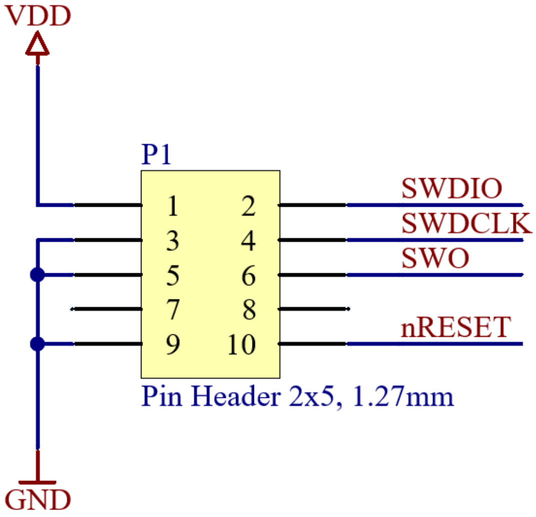


Fig. 4.7: Flash and Debug connector

4.5 PCB design

The wireless thermometer will measure the temperatures in a rotating part of a motor. Therefore the PCB shape should be a circle. The diameter of the outer motor rotor is 68 mm. To avoid imbalance, it would be best if the board would be as small and light as possible, but technological possibilities limit us. Therefore, the PCB diameter of 50 mm is chosen to compromise the PCB size, available technology, and price.

Together with the selection of the PCB size, the origin is also selected, which is placed in the geometric centre of the PCB. The main reason is to simplify the calculation of the PC balance and calibration of the position for the pick&place machine in the assembly process.

4.5.1 PCB Layer stack

Due to the small size of the PCB and the possibility of interference due to the magnetic field of the running motor, we will consider a 4 layer board design that allows to reduce the PCB to the smallest possible size and makes it easier to conduct `GND` and `VCC` potential. The following Table 4.8 shows the layer stack of the PCB design with the layers naming and thickness.

4.5.2 Electric components placement

The Bluetooth module manufacturer Laird requires to place the module on the edge, preferably the edge centre of the PCB design. The antenna should be oriented

Tab. 4.8: PCB layer stack.

Name	**Type**	**Material**	**Thickness**
Top Overlay	Overlay		8 µm
Top Solder mask	Solder mask		15 µm
Top Layer	Signal	Copper	35 µm
Dielectric 1	Prepreg	FR-4	370 µm
Inner Layer 1	GND potential	Copper	17 µm
Dielectric 2	Core dielectric	FR-4	710 µm
Inner Layer 2	VDD potential	Copper	17 µm
Dielectric 3	Prepreg	FR-4	370 µm
Bottom Layer	Signal	Copper	35 µm
Bottom solder mask	Solder mask		15 µm
Bottom Overlay	Overlay		8 µm
Total			1.6 mm

outward with *keep-out* under the antenna space.[20] Which means that there must be no signal tracks, vias or polygons under the antenna.

The battery should be placed in the centre of the PCB and should be easily changeable because there are two screws under the battery that mount the PCB on the motor shaft.

The RTD sensors should be evenly spaced. This means that the sensors can be placed on different parts of the motor rotor, and the length of the cables could be reduced. Each terminal for RTD is very close to the PCB edge. It allows connecting RTD for 4-wire measurement. On the bottom side of the PCB there is a table with correct connection of the sensors to the board. Please see the appendix D.6 on the page 97 with the bottom side overlay.

The measuring circuits MAX31865 are placed very close to the RTD terminals. The buffers and electronic load switches MIC94300 are also placed as close as possible to the corresponding MAX31865. Each MAX31865 has an LED that is connected to its power supply, and the LED is used for debugging. To reduce power consumption, these LEDs are disconnectable by soldering plates `SB1` to `SB4`.

Each MAX31865 needs a reference resistance `Rref` for correct temperature measurement. This reference resistor must be easily accessible for re-soldering. The manufacturer recommends using different reference resistors depending on the nominal value of the RTD sensor. To simplify user operation, the table on the bottom board side was created, and it is also shown in the appendix D.6.

4.5.3 Mechanical aspects

The finished wireless thermometer will be placed on the motor rotor and rotated at a speed of up to 7100 RPM. Therefore, large centrifugal forces will be imposed on the PCB and components. Therefore, it is very important to consider these facts before manufacturing and place the components on the circuit board to balance the circuit board as best as possible to avoid imbalance and potential destruction of the device.

Most of the components from the design are paired, for example, measuring ICs, buffers, load switches, most of the capacitors, etc. These components are placed exactly opposite each other, so they do not contribute to the PCB imbalance. But several components do not have a pair (Bluetooth module, debugging connector, switch and button, etc.).

4.5.4 Mass centre calculation

Unfortunately, not all electric components are paired. Therefore, the board is unbalanced, and at high speeds, there would be vibrations, force stress on the PCB, and even lead to the PCB destruction. Therefore, before manufacturing, we have to consider this aspect and calculate how much the mass centre deviates from the centre of the PCB. But first, we need to remind the basics of mechanics.

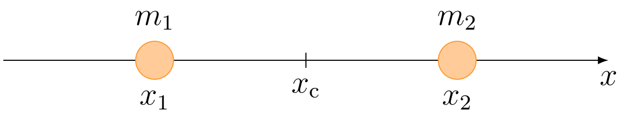


Fig. 4.8: Mass center when the $m_1 = m_2$.

For two mass points of the same weight, which are on the x-axis, it is clear that their centre or “mass centre” will be midway between them. If the x_1 and x_2 are coordinates of the points, the coordinate of the centre will be their average: [22]

$$x_c = \frac{x_1 + x_2}{2}. \tag{4.1}$$

But if, for example, the first point is more massive than the second, the point, we call the mass centre, should probably be closer to the first massive point. This is captured by the weighted average of the coordinates of the points in the equation 4.2.

$$x_c = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2}. \tag{4.2}$$

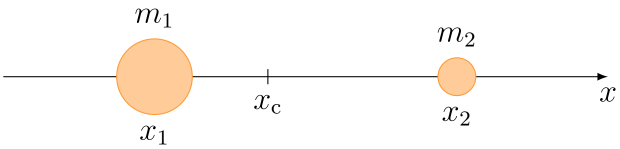


Fig. 4.9: Mass center when the $m_1 > m_2$.

The result can naturally be generalised to the three-dimensional case. [22] For position vectors we get:

$$\vec{r}_\mathrm{c} = \frac{m_1\vec{r}_1 + m_2\vec{r}_2}{m_1 + m_2}, \tag{4.3}$$

where $\vec{r}_1$ and $\vec{r}_2$ are position vectors of the mass points m_1 and m_2. The result is the position vector of the Mass centre. To generalise this relation to an arbitrarily large number of material points already is not a problem:

$$\vec{r}_\mathrm{c} = \frac{m_1\vec{r}_1 + m_2\vec{r}_2 + ... + m_N\vec{r}_N}{m_1 + m_2 + ... + m_N} = \frac{\sum\limits_{i=1}^{N} m_i\vec{r}_i}{\sum\limits_{i=1}^{N} m_i} \tag{4.4}$$

This relationship can be broken down into x and y components [22]:

$$x = \frac{\sum\limits_{i=1}^{N} m_i x_i}{\sum\limits_{i=1}^{N} m_i} \tag{4.5}$$

$$y = \frac{\sum\limits_{i=1}^{N} m_i y_i}{\sum\limits_{i=1}^{N} m_i} \tag{4.6}$$

At this point, we need to find out the weight and coordinates x and y of the individual components on the PCB and calculate the actual position of the mass centre. A complete table with all components would take too much space in this paper, so only the first few rows are shown, and the rest of the table is given in the appendix H on the page 103.

Tab. 4.9: Sample table of the components positions and weights.

Designator	Part description	weight [mg]	x [mm]	y [mm]
MOD1	BT module BL653µ	940.00	0.00	20.00
U1	MAX31865	64.00	-13.50	-12.00
C1	0402 – 1µF	1.50	8.00	11.00
R1	0603 – 430 Ω	2.00	10.50	19.00
SW1	Main switch	210.0	0.00	-21.00

After using the equations 4.5 and 4.6 we get the following position of the centre of mass:

$$\begin{aligned} x_{\mathrm{c}} &= -0.13 \text{ mm}, \\ y_{\mathrm{c}} &= 4.07 \text{ mm}. \end{aligned}$$

The useless ballast should be placed opposite of the mass centre. The coordinates of the ballast should be: $x_{\mathrm{b}} = 0.13$ mm and $y_{\mathrm{b}} = -4.07$ mm. And the weight of the ballast is calculated as:

$$m_{\mathrm{b}} = \frac{\sum\limits_{i=1}^{N} m_i \vec{r}_i}{r_{\mathrm{c}}} = \frac{\sum\limits_{i=1}^{N} m_i \vec{r}_i}{\sqrt{x_{\mathrm{c}}^2 + y_{\mathrm{c}}^2}}. \tag{4.7}$$

After the calculation, we get the following ballast position and weight if the components are placed according to the table in the appendix. The hole for placing a ballast will be drilled in the following coordinates, and the ballast will be placed there.

$$\begin{aligned} x_{\mathrm{b}} &= 0.13 \text{ mm}, \\ y_{\mathrm{b}} &= -4.07 \text{ mm}, \\ m_{\mathrm{b}} &= 6.92 \text{ g}. \end{aligned}$$

5 Software design

This chapter deals with the software part of the designated wireless thermometer. The first section 5.1 focuses on the configuration of the SPI bus, followed by section 5.2 on the control of the MAX31865 and the last section 5.3 deals with the setup of the Bluetooth LE and its implementation.

5.1 Configuration SPI

Before we start, we need to configure the SPI bus to communicate between the µC and peripheral slaves MAX31865. The SPI bus was introduced and described in the chapter 2 on the page 23.

Listing 5.1: Definitions of SPI instance from file `MAX31865.c`

```
#define SPI_INSTANCE  0             // SPI instance index
static const nrf_drv_spi_t spi = NRF_DRV_SPI_INSTANCE(
   SPI_INSTANCE);
static volatile bool spi_xfer_done;
```

First of all, we need to define the SPI instance. The used µC nRF52833 offers up to 3 SPIs instances to use [21]. We need to initialise only one SPI bus, so we define `SPI_INSTANCE` as `0` (The first possible SPI instance). The macro `NRF_DRV_SPI_INSTANCE` is called with the defined parameter `SPI_INSTANCE`. The macro creates an SPI master driver instance. And for triggering the SPI transfer completion, the flag `spi_xfer_done` is created.

5.1.1 SPI Initialisation

Before starting communication with peripherals, the SPI must be initialised. For this the function `spi_init()` is created. First of all the definitions of the `SCK`, `MISO` and `MOSI` are defined in the Listing 5.2.

Listing 5.2: Definitions of SPI pins from file `MAX31865.c`

```
#define SPI_SS_PIN       NRF_SPI_PIN_NOT_CONNECTED
#define SPI_MISO_PIN     30       // P0.30
#define SPI_MOSI_PIN     29       // P0.29
#define SPI_SCK_PIN      27       // P0.27
```

The SPI `SS` pin is set to the value `NRF_SPI_PIN_NOT_CONNECTED` which sets the value to the `0xFFFF-FFFF`. It means that this pin is not connected, and it should be pulled up and down manually in the main program.

The MISO pin is selected nRF52833 pin P0.30, which means the 30 must be written to the definition SPI_MISO_PIN. The MOSI is on the pin P.29 and the clock SCK or SCLK is set to the pin P.027. By the datasheet, [21] it is recommended to place the SPI clock to the high-frequency pin.

Listing 5.3: slave_selects_t enumeration from file MAX31865.c

```
typedef enum {
  SLAVE_1 = 23,     // P0.23
  SLAVE_2 = 24,     // P0.24
  SLAVE_3 = 17,     // P0.17
  SLAVE_4 = 31      // P0.31
} slave_selects_t;
```

The previous Listing 5.3 shows the enumeration of slave select pins. Each slave select is connected to the one of the MAX31865 IC. The slave select could also be known as chip select. Then we can make the spi_init() function.

Listing 5.4: Function spi_init() from file MAX31865.c

```
void spi_init(void)
{
   nrf_drv_spi_config_t spi_config = NRF_DRV_SPI_DEFAULT_CONFIG;

   spi_config.ss_pin      = NRF_SPI_PIN_NOT_CONNECTED;
   spi_config.miso_pin    = SPI_MISO_PIN;
   spi_config.mosi_pin    = SPI_MOSI_PIN;
   spi_config.sck_pin     = SPI_SCK_PIN;

   spi_config.mode        = NRF_DRV_SPI_MODE_1;
   spi_config.frequency   = NRF_DRV_SPI_FREQ_4M;
   spi_config.bit_order   = NRF_DRV_SPI_BIT_ORDER_MSB_FIRST;

   APP_ERROR_CHECK(nrf_drv_spi_init(&spi, &spi_config,
      spi_event_handler, NULL));
}
```

The Listing 5.4 shows the spi_init() function which initialise the SPI bus on the nRF52833 µC. In the first step, the default configuration is written to the spi_config. In the next step, the communication pins are selected. The MAX31865 needs to communicate in SPI modes 1, or 3 [5]. The SPI mode 1 is selected. Maximum allowed SCLK frequency from MAX31865 is 5 MHz. The SCLK frequency of the communication is set to 4 MHz. Also, the bit order is selected to MSB first.

The last macro APP_ERROR_CHECK calling error handler function if supplied error code any other than NRF_SUCCESS. The function nrf_drv_spi_init() initialises the SPI.

5.1.2 SPI event handler

The `spi_event_handler` is called after successful spi transfer and change the flag `spi_xfer_done` from `false` to `true`. It is important during the SPI transfer. Because the flag `spi_xfer_done` does not allow a new SPI transfer until the actual is not finished.

Listing 5.5: Function `spi_event_handler()` from the file `MAX31865.c`

```
void spi_event_handler(nrf_drv_spi_evt_t const * p_event,
                       void *                    p_context)
{
    spi_xfer_done = true;
}
```

5.1.3 Reading Register

The MAX31865 communicates with the µC via internal registers which are described in the subsection 5.2.1 on the page 57. The following function `readRegisterN()` on the Listing 5.6 reads N-bit register from the MAX31865. Parameters of this function are: `addr` for transferring address code. The `*buffer` is a pointer to the buffer where the received register will be stored. `n` sets the length of the pointed buffer and finally the `slave_select` selects the slave which with which communication has to be established.

Listing 5.6: Function `readRegisterN()` from file `MAX31865.c`

```
void readRegisterN( uint8_t addr,
                    uint8_t *buffer,
                    uint8_t n,
                    slave_selects_t slave_select)
{
   addr &= 0x7F;            // make sure top bit is not set

   nrf_gpio_pin_clear(slave_select);
   spi_xfer_done = false;

   APP_ERROR_CHECK(nrf_drv_spi_transfer(&spi,&addr,1,buffer,n));

   while (!spi_xfer_done)
   {
      __WFE();
   }

   nrf_gpio_pin_set(slave_select);
}
```

To make sure the top bit is not set, a bitwise AND is performed to confirm that the MAX31865 is switched to reading mode. The SS pin is set to logic low and the flag spi_xfer_done is set to false before starting the transfer. The while loop is entered during the transferring data and waits until the transfer is completely done. After successfully transfer, the received data are stored in the buffer, and the SS pin is set to logic high.

Function readRegister8() and readRegister16() are based on the described function readRegisterN() that read 8 or 16-bit registers respectively. The code of this functions is very similar and it is not discussed in detail here, but it is included on the attached CD with other attachments I

5.1.4 Writing Register

After reading the registers, we want to change the settings, and therefore we have to write new registers. The following function writeRegister8() write 8-bit register to the MAX31865. Parameters of this function are: addr for transferring write code. The data stores the value of the writing register, and the slave_select selects the slave which with which communication has to be established.

Listing 5.7: Function writeRegister8() from file MAX31865.c

```c
void writeRegister8(uint8_t addr,
                    uint8_t data,
                    slave_selects_t slave_select)
{
    addr |= 0x80; // make sure top bit is set

    uint8_t buffer[2] = {addr, data};

    nrf_gpio_pin_clear(slave_select);

    spi_xfer_done = false;
    APP_ERROR_CHECK(nrf_drv_spi_transfer(&spi, buffer, 2, NULL, 0));

    while (!spi_xfer_done)
    {
        __WFE();
    }
    nrf_gpio_pin_set(slave_select);
}
```

To ensure the top bit is set, a bit sum is performed to confirm that the MAX31865 is switched to writing mode.

The `addr` and `data` are stored in `buffer`. The `SS` pin is set to logic low and the flag `spi_xfer_done` is set to `false` before starting the transfer. During the transferring data the while loop is entered and wait until the transfer is completely done. After successfully transfer the received data are stored in the buffer and the `SS` pin is set to logic high.

5.2 Controlling MAX31865

The section 4.2 on the page 40 describes the MAX31865 from its hardware perspective. However, this section focuses on the description of the software control and communication µC with MAX31865 via registers which are described in the datasheet [5]. The main inspiration for controlling the MAX31865 is from the Arduino C++ library from Adafruit, which is accessible on the [23]. This library has been rewritten to the C and modified for use 4 sensors measurement on Nordic nRF52833.

5.2.1 Internal registers

Communication between the µC and MAX31865 is performed using the SPI bus using 8-bit internal registers, which contain status, configuration data and a register. Internal registers are accessed using `0x0X` address for reading and `0x8X` address for writing, where the “`X`” represents the hexadecimal value of the register. Data is read from or written to the registers MSB first.

Tab. 5.1: Configuration register definition. [5]

D7	D6	D5	D4	D3	D2	D1	D0
V_{BIAS} 1 = ON 0 = OFF	Conversion mode 1 = Auto 0 = Normally off	1-shot 1 = 1–shot (auto clear	3-wire 1 = 3–wire 0 = 2 or 4 wire	Fault Detection Cycle Control		Fault Status Clear 1 = Clear (auto-clear)	50/60 Hz filter select 1 = 50 Hz 0 = 60 Hz

Configuration register – `0x00`

When the configuration register `0x00` is transferred from µC to the MAX31865. The IC returns the actual configuration of the registers. The configuration bits are described in the following paragraphs:

BIAS (D7)

BIAS may be disabled to reduce power dissipation when the conversion is not performed. The measuring current is not flowing throw the RTD sensor. Write `1` for turning on the BIAS before beginning a single (1-Shot) conversion. During the automatic conversion, the BIAS remains on continuously.

Conversion mode (D6)

Write `1` to select automatic conversion mode, which starts conversion at a 50/60 Hz rate. (It depends on the value of the D0 bit.) Write `0` to this bit and enter the "Normally Off" mode. The 1-Shot conversion may be initialised from this mode.

1-Shot (D5)

When the conversion mode is "Normally Off" and the D6 bit is set to `0` write `1` to start a single conversion. The conversion is triggered when `CS` (or `SS`) pin goes high after writing `1` to this bit. The measuring is valid if the BIAS is turned on. The single conversion requires approximately 52 ms in 60 Hz filter mode and 62.5 ms in 50 Hz mode. After conversion, the D5 bit is cleared.

Number of wire (D4)

Write `1` to this bit when the 3-wire measurement of the RTD is used. In this mode the voltage between `FORCE+` and `RTDIN+` is subtracted from `RTDIN+` − `RTDIN-` to compensate error caused by using only single wire for the `FORCE-` and `RTDIN-` connection. When using 2 or 4-wire connection, write `0` to this bit.

Listing 5.8: Definitions of `numwires_t` enumeration from file `MAX31865.h`

```c
typedef enum {
  MAX31865_2WIRE = 0,
  MAX31865_3WIRE = 1,
  MAX31865_4WIRE = 0
} max31865_numwires_t;
```

Fault detection cycle (D3-D2)

The fault detection cycle checks for three faults by making the following voltage comparisons and setting the associated bits in the Fault Status Register:

- Is the voltage at `REFIN-` $>$ than 85 % $\cdot V_{\text{BIAS}}$?
- Is the voltage at `REFIN-` $<$ than 85 % $\cdot V_{\text{BIAS}}$
  when `FORCE-` input switch is open?

- Is the voltage at RTDIN- $<$ than 85 $\% \cdot V_{\mathrm{BIAS}}$
  when FORCE- input switch is open?

Fault status clear (D1)

Write 1 to this bit while writing 0 to bits D5, D3, and D2 to clear all fault status bits. After clearing faults, the bit D1 is cleared to the 0.

50/60 Hz (D0)

This bit selects notch frequencies for the noise rejection filter. Write 1 to this bit to reject 50 Hz frequency and its harmonics. For filtering 60 Hz, write 0.
Note: It is not allowed to change notch frequencies in auto conversion mode.

The following code listing 5.9 show the different definitions for each configuration register.

Listing 5.9: Definitions of configuration registers from file MAX31865.h

```
#define MAX31865_CONFIG_REG        0x00
#define MAX31865_CONFIG_BIAS       0x80
#define MAX31865_CONFIG_MODEAUTO   0x40
#define MAX31865_CONFIG_MODEOFF    0x00
#define MAX31865_CONFIG_1SHOT      0x20
#define MAX31865_CONFIG_3WIRE      0x10
#define MAX31865_CONFIG_24WIRE     0x00
#define MAX31865_CONFIG_FAULTSTAT  0x02
#define MAX31865_CONFIG_FILT50HZ   0x01
#define MAX31865_CONFIG_FILT60HZ   0x00
```

5.2.2 Initialisation

For the right initialisation of MAX31865, we have to make several necessary settings:

1. We have to set the number of measuring wires used to measure the RTD sensor.
2. BIAS is set to false. This means that there will not flow any current through the RTD. It's to reduce the sensor self-heating and also to save the power.
3. The Conversion mode register is set to false to disable auto conversion, again to save power.
4. Finally, the faults from the previous session will be cleared, if any remain in memory.

Listing 5.10: Function `MAX31865_init()` from file `MAX31865.c`

```c
bool MAX31865_init(max31865_numwires_t wires,
                   slave_selects_t slave_select)
{
    setWires(wires, slave_select);
    enableBias(false, slave_select);
    autoConvert(false, slave_select);
    clearFault(slave_select);

    NRF_LOG_INFO("Initialisation of MAX31865 was successful");
    return true;
}
```

To simplify initialisation, the function `MAX31865_init()` was created, which has two parameters: `wires` and `slave_select`. The parameter `wire` is defined as enumeration `max31865_numwires_t` in the file `MAX31865.h`. Using the parameter `wire` we set up the IC to 2, 3 or 4-wire measurement. Parameter `slave_select` sets up communication with the specific slave. The designing wireless thermometer has four sensors, therefor the parameter for selecting the slave is necessary to select right sensor.

After the successful initialisation, the message *"Initialisation of MAX31865 was successful"* is printed by `NRF_LOG_INFO()` to the UART.

5.2.3 Setting the measuring wires

The function for setting the number of measuring wires is called `setWires()` and does not return any value. It has two parameters: `wires` and `slave_select`. The both parameters are described in the previous function `MAX31865_init()` in the subsection 5.2.2.

Listing 5.11: Function `setWires()` from file `MAX31865.c`

```c
void setWires(max31865_numwires_t wires,
              slave_selects_t slave_select)
{
  uint8_t t = readRegister8(MAX31865_CONFIG_REG, slave_select);
  if (wires == MAX31865_3WIRE)
  {
    t |= MAX31865_CONFIG_3WIRE;
  }

  else
  {
    // 2 or 4 wire
    t &= ~MAX31865_CONFIG_3WIRE;
  }

  writeRegister8(MAX31865_CONFIG_REG, t, slave_select);
}
```

The first step of the `setWires()` function is the initialisation `uint8_t` variable `t`. The function `readRegister8()` is called with two parameters: `MAX31865_CONFIG_REG` and `slave_select`. The IC receives configuration register and immediately sends back the answer with actual registers settings of IC. This value is now stored as `t`.

The next step verifies if the 3 or 2, or 4-wire measurement is selected. If the 3-wire is selected the 4th bit of the register `t` is changed from `0` to `1`. If the 2 or 4-wire measurement is selected, the fourth bit of the register `t` is changed to `0`.

The last step of the function `setWires()` writes the new changed configuration register `t` to the IC. The function `writeRegister8()` is called with three parameters: `MAX31865_CONFIG_REG`, `t` and `slave_select`.

5.2.4 Read faults

The MAX31865 can detect multiple faults. Some faults are detected on every conversion, while others are detected only when the master requests the fault detection cycle. The result of the fault detection cycle is written on the D3 and D2 bits in the register. The function `readFault()` reads the fault status from the MAX31865.

Listing 5.12: Function `readFault()` from file `MAX31865.c`

```c
uint8_t readFault(slave_selects_t slave_select)
{
    return readRegister8(MAX31865_FAULTSTAT_REG, slave_select);
}
```

The function readFault() returns the value of the fault. The value could be compared with fault status table of definitions which are defined in MAX31865.h file.

Listing 5.13: List of fault status values from file MAX31865.h

```
#define MAX31865_FAULT_HIGHTHRESH 0x80
#define MAX31865_FAULT_LOWTHRESH  0x40
#define MAX31865_FAULT_REFINLOW   0x20
#define MAX31865_FAULT_REFINHIGH  0x10
#define MAX31865_FAULT_RTDINLOW   0x08
#define MAX31865_FAULT_OVUV       0x04
```

5.2.5 Read RTD resistance

The important function in the designated wireless thermometer is the readRTD() function which return the value of ADC conversion. This function has only one parameter slave_select which selects the slave sensor and requests it for the conversion. The following Listing 5.14 captures the read ADC value process. The description is on the next page.

Listing 5.14: Function readRTD() from file MAX31865.c

```
uint16_t readRTD(slave_selects_t slave_select)
{
    clearFault(slave_select);
    enableBias(true, slave_select);

    nrf_delay_ms(10);

    uint8_t t;
    t = readRegister8(MAX31865_CONFIG_REG, slave_select);
    t |= MAX31865_CONFIG_1SHOT;

    writeRegister8(MAX31865_CONFIG_REG, t, slave_select);

    nrf_delay_ms(65);

    uint16_t rtd;
    rtd = readRegister16(MAX31865_RTDMSB_REG, slave_select);

    // Disable bias current to reduce selfheating of PT100.
    enableBias(false, slave_select);

    rtd >>= 1;

    return rtd;
}
```

First of all, the function `readRTD()` clears the faults and enables BIAS. However, the manufacturer recommends waiting for measuring RTD value of about 10 ms to stabilise transients. Then the function writes a register to enable 1-shot conversion. The 1-shot conversion takes about 62.5 ms [5] therefore, the delay 65 ms is inserted. After a recommended time, the ADC conversion value is read. When the transfer of RTD value is finished, the BIAS is turned off, and the function returns the measured ADC value of the RTD sensor.

5.2.6 Read temperature

The main function of the designated wireless thermometer is reading temperature function simply called `readTemperature()`. This function has three input variables `RTDnominal`, `refResistor` and `slave_select`. The `RTDnominal` is defined in the definition table at the top of `main.c` file. This variable refers the nominial RTD value at 0°C. The next `refResistor` contains a value of the reference resistor (`R1`–`R4`). And last variable `slave_select` selects the sensor.

The function is simple, but it contains a lot of equations, so it would be unclear here, so only code snippets are listed.

Listing 5.15: Real RTD resistance value from `readTemperature()` in `MAX31865.c`

```
1      Rt = 0;
2      Rt = readRTD(slave_select);
3      Rt = Rt / 32768;
4      Rt = Rt * refResistor;
```

First of all we have to get real resistance value of RTD sensor. The Listing 5.15 shows the easy calculation resistance from ADC conversion value which is returned from `readRTD()` function. On the third line the number `32768` is maximum value range from 15-bit ADC ($2^{15} = 32768$). The variable `Rt` contains the measured value from RTD sensor and it corresponds with R_ϑ in equations.

You may notice in the section 1.3 on the page 17 the Callendar-Van Dusen equations have two different forms. The first one (1.3) is for temperatures below zero and the second (1.4) is for positive temperatures. The second equation is a simple quadratic equation, and it is easy to solve it and get the temperature. But the first Equation 1.3 contains the temperature ϑ to the fourth power, and it is too complex to solve this form of equations. But there is a way how to get the result – we can use successive approximation [24]:

$$\vartheta_1 = \frac{\frac{R_\vartheta}{R_0} - 1}{A + 100 \cdot B}, \tag{5.1}$$

$$\vartheta_{n+1} = \vartheta_n + \frac{1 + A\vartheta_n + B\vartheta_n^2 + C\vartheta_n^3 \cdot (\vartheta_n - 100) - \frac{R_\vartheta}{R_0}}{A + 2B\vartheta_n - 300C\vartheta_n^2 + 4C\vartheta_n^3}. \tag{5.2}$$

The article [24] says it only takes two iterations for sufficient accuracy. The result of the temperature is returned by the `readTemperature()` function. If the temperature is greater than zero, the calculation will be simplified, and only the quadratic equation will be solved. This calculation is so simple that we will not bore the reader with simple calculations.

5.3 Bluetooth implementation

This section does not aim to describe all necessary functions for the correct running of a BLE application but rather describes the most important functions directly involved in writing the measured temperature values to the BLE characteristics.

5.3.1 BLE parameter settings

BLE parameters settings in the `main.c` code is done in the source file `main.c`, which is located in the main folder. All throw parameter values are defined by the `#define` command at the top of the source code. Just overwrite the corresponding default value to the desired value to change them.

Listing 5.16: Main BLE definitions from `main.c` file

```
#define DEVICE_NAME                            "Remote_Thermometer"
#define MANUFACTURER_NAME                      "Vladimir_Pokorny"
#define APP_ADV_INTERVAL                       160

#define APP_ADV_DURATION                       6000
```

Definitions `DEVICE_NAME` and `MANUFACTURER_NAME` define the name of device and name of a manufacturer. They are used to identify the device in a device list during pairing. The `APP_ADV_INTERVAL` is the advertising interval. The number of `APP_ADV_INTERVAL` is in units in 0.625 ms. The value 160 corresponds to a 100 ms advertising interval. A larger advertising interval causes greater BLE power consumption in the advertising state. Following definition `APP_ADV_DURATION` sets the duration advertising state. After this time, the BLE should turn off and wait for the reset button press to start advertising broadcasting again.

5.3.2 BLE service

Before we begin creating a new BLE service, we need first to define the UUID. (The purpose and reason why the UUID is used were discussed in the subsection 3.3.6 on the page 35.) The used UUID in big-endian is:

`0000ABCD-5468-616E-6B73-4D6F6D446164,`

while we use the little-endian representation in our project. Thus, we have to reverse the byte-ordering and define the `BLE_UUID_OUR_BASE_UUID`:

Listing 5.17: The UUID service definition from `temperature_service.h` file

```
#define BLE_UUID_OUR_BASE_UUID                                    \
            {0x64, 0x61, 0x44, 0x6D, 0x6F, 0x4D, 0x73, 0x6B, \
             0x6E, 0x61, 0x68, 0x54, 0x00, 0x00, 0x00, 0x00}

#define BLE_UUID_OUR_SERVICE        0xABCD
```

After defining the service UUID, we can create the new Bluetooth Low Energy service by `our_service_init()` function. At this point the observer can find and connect to the device and see the empty service:

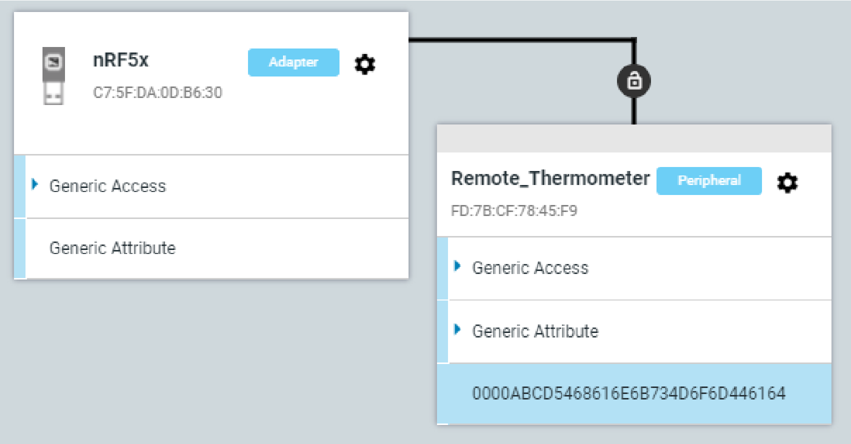


Fig. 5.1: The nRF Connect from Nordic Semiconductor shows the BLE service. The application could be downloaded on [25].

5.3.3 BLE characteristics

As was described in subsection 3.3.7 on the page 36, the BLE characteristic is used to exchange data between BLE devices. In the case of our wireless thermometer, each characteristic transmits data from just one temperature sensor. Before declaring the functions for initialising the characteristics, we must define the UUIDs for the individual sensor characteristics. The UUIDs are 16-bit values and could be chosen from the predefined table from Bluetooth SIG: [10], but it also could be a random value. In our case the first sensor has number 1, the second 2 and so on. The 16-bit of UUIDs values in little-endian are:

Listing 5.18: The BLE characteristics UUIDs from temperature_service.h file

```
#define BLE_UUID_1_TEMPERATURE_UUID     0x1000       // Sensor 1
#define BLE_UUID_2_TEMPERATURE_UUID     0x2000       // Sensor 2
#define BLE_UUID_3_TEMPERATURE_UUID     0x3000       // Sensor 3
#define BLE_UUID_4_TEMPERATURE_UUID     0x4000       // Sensor 4
```

After the UUIDs definition, we can add new characteristics to our temperature service, which was created in the previous subsection. The special functions our_char_add_1() to our_char_add_4() create and add our characteristics and their properties into created service. Only the most important parts of this function are highlighted.

read/write permissions

Our temperature characteristics transfer data from the sensor to the central device (laptop, PC or smartphone). Therefore, allowing the central device to overwrite the measured values makes no sense. Only the read permission is allowed:

Listing 5.19: The characteristic r/w permissions from temperature_service.c

```
    char_md.char_props.read = 1;
    char_md.char_props.write = 0;
```

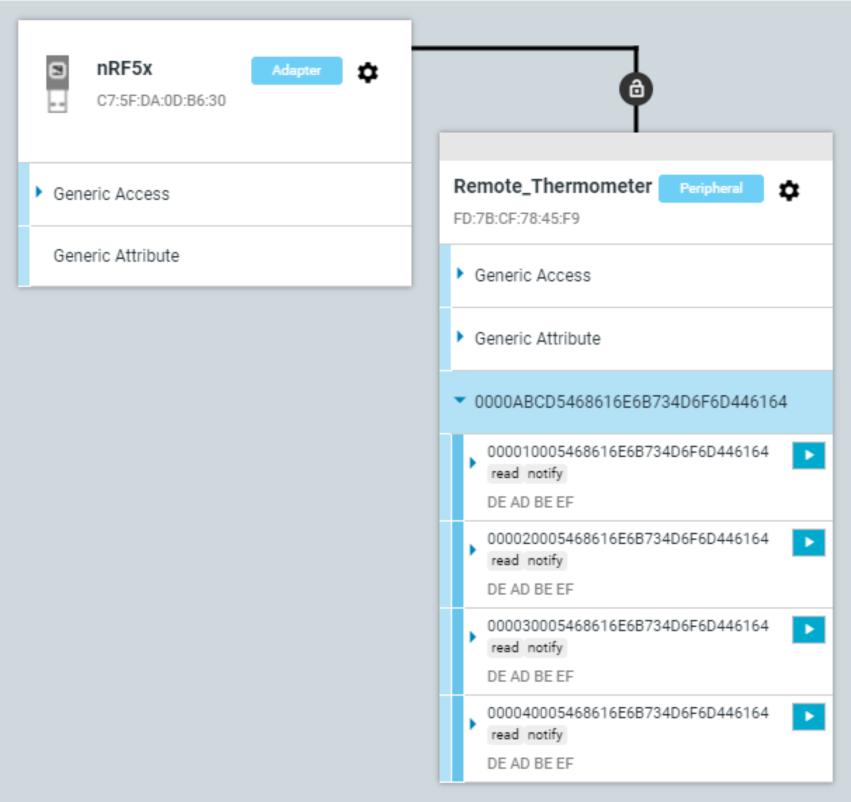
Initial data length and value

Next Listing 5.20 shows the definitions of the maximal and initial length of the data value. The initial value is set to 0xDE,0xAD,0xBE,0xEF. The DEADBEEF is debugging value and in our wireless thermometer application indicates the non-updating characteristic or the sensor is not enabled.

Listing 5.20: The init. length and value of BLE char. from temperature_service.c

```
    // Set characteristic length in number of bytes
    attr_char_value.max_len     = 4;
    attr_char_value.init_len    = 4;
    uint8_t value[4]            = {0xDE,0xAD,0xBE,0xEF};
    attr_char_value.p_value     = value;
```

After settings permission and initial values, the characteristic could be added to the temperature service characteristic. The central device can see the following data structure:



nRF5x
Adapter
C7:5F:DA:0D:B6:30

Generic Access

Generic Attribute

Remote_Thermometer
Peripheral
FD:7B:CF:78:45:F9

Generic Access

Generic Attribute

0000ABCD5468616E6B734D6F6D446164

00001000546816E6B734D6F6D446164
read notify
DE AD BE EF

00002000546816E6B734D6F6D446164
read notify
DE AD BE EF

00003000546816E6B734D6F6D446164
read notify
DE AD BE EF

00004000546816E6B734D6F6D446164
read notify
DE AD BE EF

Fig. 5.2: The nRF Connect from Nordic Semiconductor shows the BLE characteristics. The application could be downloaded on [25].

5.3.4 Update temperature values function

After the successful reading temperature, the characteristic should be updated, and the connected central device should be notified that the characteristic was updated. The function `our_temperature_characteristic_update()` is created. It has only two parameters: pointer `*p_our_service` which points to the structure with the BLE service. The second parameter is pointer `*temperature_value` which points to the variable where the actual temperature is stored.

The update characteristic function updates the corresponding characteristic, and it also sends a notification that the characteristic was updated. The notification and other ATT commands are described in subsection 3.3.6 on the page 35.

5.4 Table of definitions in main.c

At the beginning of the main.c file, there is a table with properties of the measuring, and only these values should be changed by the user.

RTD nominal value

The definition RTD_R0 defines the nominal value of the RTD sensor at 0°C. The RREF defines the value of the reference resistor (R1–R4). The range of the reference resistor value should be from 350 Ω to 10 kΩ [5].

Read temperature interval

The definition READ_TEMPERATURE_INTERVAL sets the measuring interval. The minimum save measuring interval should be 350 ms for 4 sensor measuring. (Each sensor takes around 80 ms to read temperature).

Sensor enable

The next definition SENSOR_ENABLE has only two possible states: true or false. If it is true the sensor is enabled and used for reading temperature. However, if it is false the sensor is turned off and saves the energy.

Number of sensor wires

The last definition NUMWIRE_SENSOR sets the number of used measuring wires. It has only three states: MAX31865_2WIRE which sets the 2-wire measurement, and it is the cheapest, MAX31865_3WIRE sets the 3-wire measurement, and the last MAX31865_4WIRE sets the 4-wire measurement.

Listing 5.21: The table of definitions from file main.c

```c
#define RTD1_R0 (double)             100.0       // Ohm
#define RREF1 (double)               430.0       // Ohm

#define READ_TEMPERATURE_INTERVAL    1000        // ms

#define SENSOR_1_ENABLE              true
                                 //  false

#define NUMWIRE_SENSOR_1             MAX31865_4WIRE
                                 //  MAX31865_2WIRE
                                 //  MAX31865_3WIRE
```

Previous Listing 5.21 is abbreviated and only shows the definitions for the first sensor.

5.5 Main loop

The main loop is divided into two parts: *Initialisation part* and *Measuring part.* The first initialisation is used to set the GPIO ports

5.5.1 Initialisation

Before we start with reading temperatures, we have to set up the application's necessary components. First of all we have to initialise GPIO pins such as `EN_1` to `EN_4` and `SLAVE_1` to `SLAVE_4` which are set to output pins.

Next, we have to initialise the SPI bus. We just can call the `spi_init()` function which we discussed in the subsection 5.1.1. When the SPI bus is successfully initialised, the slaves MAX31865 can be configured, and the `MAX31865_init()` function is called. Each slave is configured with the number of measuring wires which is set in the table of definitions.

After initialisation of SPI and MAX31865, the BLE could be established, and all necessary functions are called, such as `ble_stack_init()` which initialises the SoftDevice and the BLE event interrupt, `gatt_init()` which is a function for initialising the GATT module. The BLE service and its characteristics are initialised by `services_init()` and finally, The device starts with BLE advertising when the function `advertising_start()` is called.

5.5.2 Measuring part

The measuring part is more interesting because temperature measurements and characteristic updates are used. The Listing 5.22 is shortened and shows the main measuring loop only for the first sensor.

When the program enters the infinity loop the `idle_state_handle()` puts the processor to sleep mode and waits for a timer that sets the `read_temperature_flag` to true every time the measurement interval is triggered.

If the first sensor is enabled the GPIO pin `EN_1` is pulled up and the MAX31865 is turned on. There is a small delay (2 ms) to wait to stabilise the voltage level and transients. Then the temperature reading starts and the `float` value is stored in `RTD_Temperature_1` variable. The BLE characteristic accepts only the `int` value, therefor the `RTD_Temperature_1` is multiplied by `1000` and the new value is stored in the `int_temperature_1` variable. The MAX31865 is turned off and the characteristic is updated with new temperature value. The loop continuous with next sensors and in the end of the loop the `read_temperature_flag` is set to `false` and the program continues from the beginning.

Listing 5.22: The main measuring part from `main.c` file

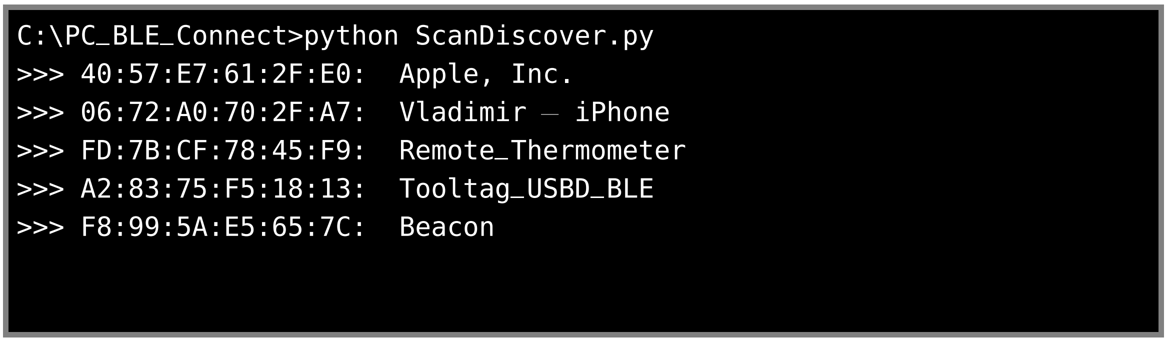
```c
while (true)
{
  idle_state_handle();

  if (read_temperature_flag == true)
  {
    if (SENSOR_1_ENABLE == true)
    {
      // 1. Temperature sensor
      nrf_gpio_pin_set(EN_1);
      nrf_delay_ms(2);

      RTD_Temperature_1 = readTemperature(RTD1_R0, RREF1,
         SLAVE_1);
      int_temperature_1 = RTD_Temperature_1 * 1000;

      nrf_gpio_pin_clear(EN_1);
      our_temperature_characteristic_update_1(&m_our_service,
         &int_temperature_1);
    }
    read_temperature_flag = false;
  }
}
```

5.6 PC applications

A wireless thermometer alone would be useless unless there were also a computer or mobile phone app to store the measured data. Therefore the easy to use app for the PC is created. It is written in Python and uses the `bleak` library, which supports reading, writing and getting notifications from GATT servers, as well as a function for discovering BLE devices. [26] The two Python scripts were written.

5.6.1 Scan & Discover app

This Python script scans and prints a list of all the available BLE devices. The scan time takes 5 seconds, and after it, the list is printed in a terminal. This application is important to find out the unique MAC address of the wireless thermometer.



```
C:\PC_BLE_Connect>python ScanDiscover.py
>>> 40:57:E7:61:2F:E0:  Apple, Inc.
>>> 06:72:A0:70:2F:A7:  Vladimir – iPhone
>>> FD:7B:CF:78:45:F9:  Remote_Thermometer
>>> A2:83:75:F5:18:13:  Tooltag_USBD_BLE
>>> F8:99:5A:E5:65:7C:  Beacon
```

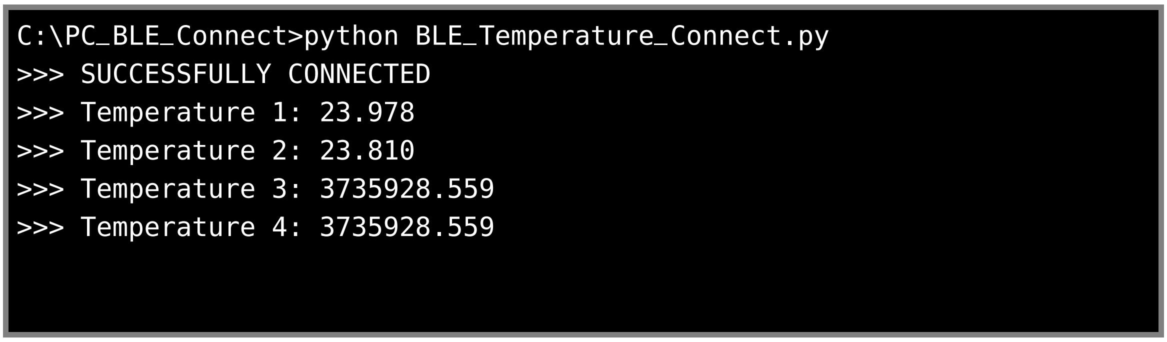
Fig. 5.3: Output from `ScanDiscover.py` application

For example, the nRF Connect for Mobile app, which is available for both iOS and Android, can serve the same purpose. This can scan BLE devices, connect to them and readout characteristics if the device security allows it. Furthermore, since the designated wireless thermometer uses open communications, nRF Connect can even connect to it and detect other attributes such as UUID codes service and characteristics and the MAC address.

5.6.2 BLE Temperature Connect app

It is the Python script which connects to the BLE device, reads temperature values, prints and writes the measured values into the `YYYY-MM-DD-HH-MM_temperature.csv` file. Before printing and writing, the downloaded values are divided by 1000. The characteristics are updated by the temperature value multiplied by 1000 to keep decimals. The output of the terminal is shown in the Figure 5.4, and it is adjusted to human-readable values. Each temperature value is printed on the Celsius scale. You might notice the values `3735928559` which corresponds with the 32-bit value `0xDEAD`

BEEF. The 3rd and 4th temperature sensors are turned off, or their characteristics are not updated.



```
C:\PC_BLE_Connect>python BLE_Temperature_Connect.py
>>> SUCCESSFULLY CONNECTED
>>> Temperature 1: 23.978
>>> Temperature 2: 23.810
>>> Temperature 3: 3735928.559
>>> Temperature 4: 3735928.559
```

Fig. 5.4: Output from BLE_Temperature_Connect.py application

This application is used in the next chapter to verify the functionality of the wireless thermometer.

The next step in developing this product could be creating a GUI application that could use the BLE Temperature Connect app as a back-end app. For example, the GUI application could plot a graph in real-time, and also the wireless thermometer could also be set through the GUI app.

6 Device commissioning

This chapter describes the final PCB assembly. The initial electric test of the board is in the section 6.2. The firmware flashing is discussed in the section 6.3 and the test of the BLE and sensors is described in the section 6.4. And finally, this chapter ends with accuracy measurements in the section 6.7.

6.1 Assembly PCB and components

The PCB was manufactured in the Würth Elektronik GmbH company. The boards for six devices were made because it is better to make more than one prototype. The panel of six boards was ordered an it can be seen in the Figure 6.1.

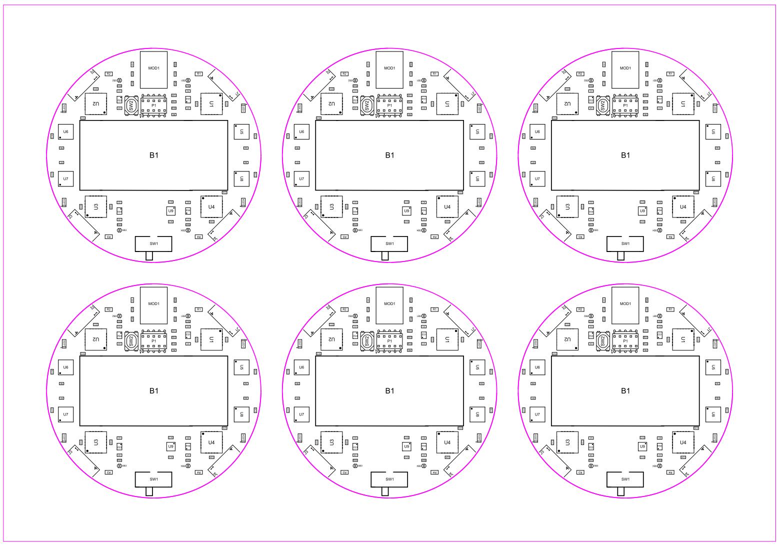


Fig. 6.1: PCB panel of six boards.

The Figure 6.2 captures microscope view to the assembled PCB with a ruler. You may notice the size of the smallest pad is smaller than 0.5 × 0.5 mm. Therefor the pick&place machine was used for the precise assembly. After full assembly the board panel was carefully put to reflow soldering process.

Reflow soldering

The thermal profile was set according to the manufacturers recommendations. The BT module Laird BL653µ is the most sensitive component to the reflow process. It should not be above 260°C and not for more than 30 seconds. The manufacturer also recommends to solder module only once, otherwise the BL653µ internal components can be impacted by soldering.

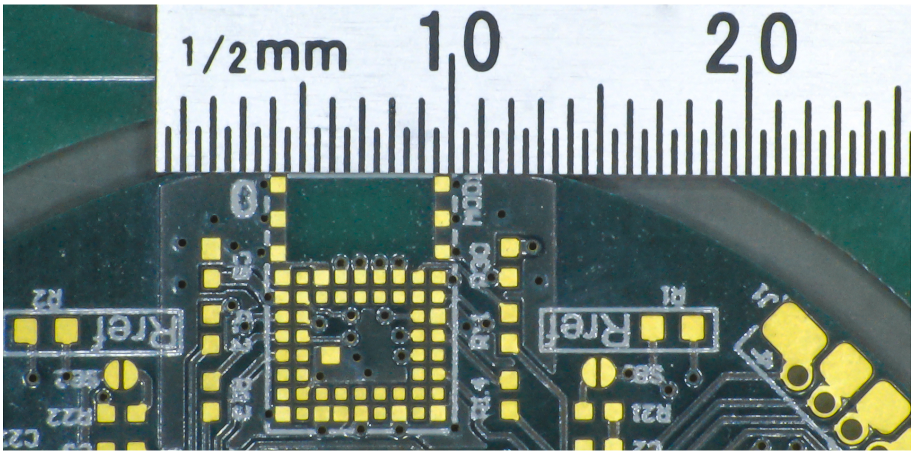


Fig. 6.2: Detail of the unassembled PCB.

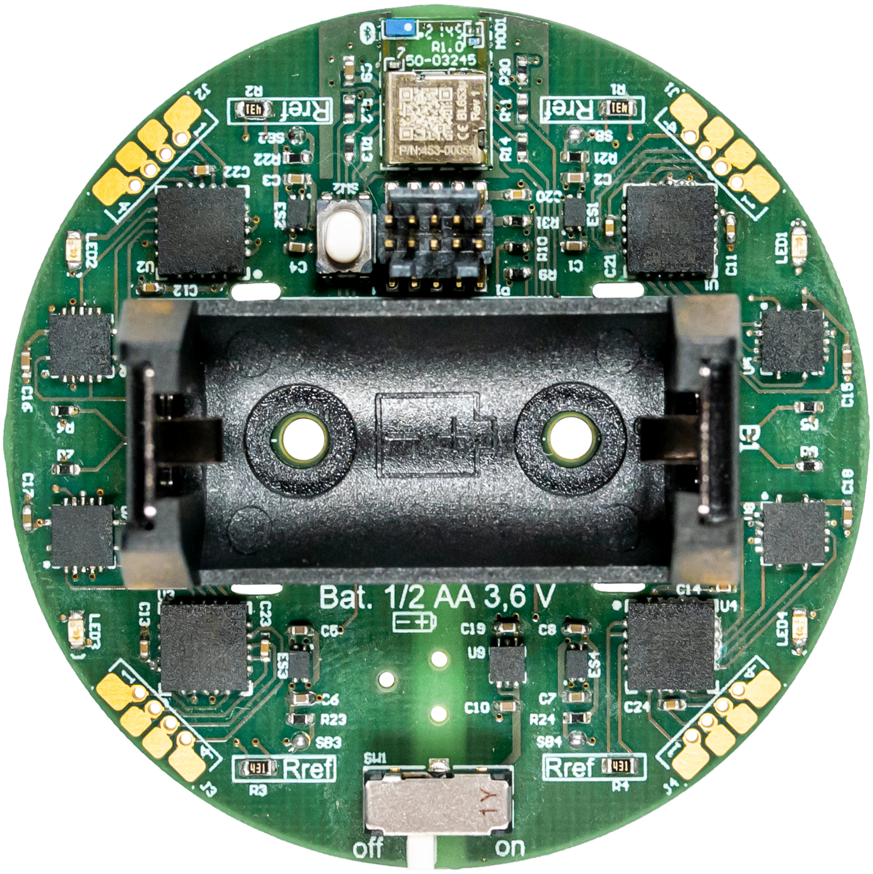


Fig. 6.3: Assembled one PCB.

6.2 Initial electric test

After the successful reflow soldering the optical inspection was carried out and any visible short-circuits were resolved. The short-circuits between the `VDD` and `GND` were checked and all errors were resolved. After ensuring that there is no short circuit, the first electrical test was carried out by connecting the 3.6 V supply voltage level to the terminals of battery holder. A current limit of 40 mA was set on the laboratory power supply. The current limitation was set to protect circuits from destruction. During this test the current should not rise above 10 mA or the voltage should not fall from 3.6 V. Otherwise there would be a mistake in the form of a short-circuit or broken component.

The initial state of the GPIO pins according the Laird datasheet [20] is high. So, the next step is to short the solder bridges (`SB1`–`SB4`) and check the right polarity of LEDs. Each LED should be glowing brightly, if one is not glowing or is not glowing enough, there may be a wrong LED polarity, a fault in the contact corresponding `EN` net with BT module `MOD1`. Or also the corresponding load switch (`ES`) may be damaged.

6.3 Firmware flash

The firmware flashing is very easy. Just only the recommended J-link programmer can be used. Or also the development kit can be used to flash firmware to the custom board. If the development kit nRF52833 DK from Nordic is used the *Debug out* connector `P19` is used. The custom board is connected by 10-pin SWD cable with 1.27 mm pitch of the contacts.

6.4 BLE and sensor test

After successful flashing the firmware into the μC, the BLE and sensors were checked.

6.4.1 BLE test

Before starting the BLE testing, all sensors were disabled in firmware to test the correct BLE functionality first. When the device is running, each debug LED should be turned off. And the device appeared in the nRF Connect from Nordic Semiconductor. The Figure 6.4 captures the screen from nRF Connect with connected device *Remote_Thermometer* and its MAC address. You can notice the correct service the bellow of the list of BLE preferences. The service contained the four characteristics with the initial value `0xDEAD BEEF` which was expected.

6.4.2 Sensor test

After the successful BLE connection, the sensor test was proceeded. The new firmware was uploaded into the device. Then the device was connected with com-

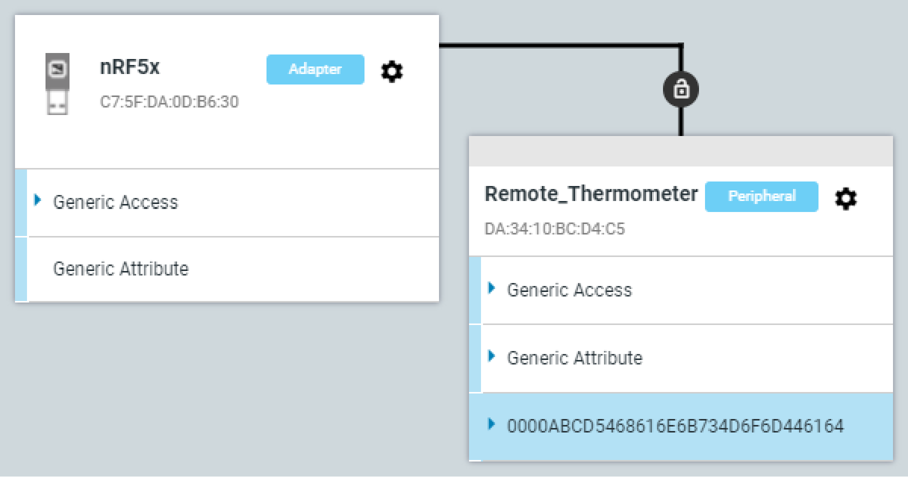


Fig. 6.4: Bluetooth Low Energy test.

puter and the characteristics transferred the value `0x9C-4E-FC-FF` which corresponds with issue in characteristics update or with the problem with sensors and reading temperatures. The problem was debugged and after short time it was found that the problem in the tested sample is a bad contact of the `SCLK` signal of SPI bus. The logic probe was used to measure the logic levels of the SPI bus. The result is in the Figure 6.5. The figure captures the problem in the `SCLK` line. There is open circut with `SCLK` pin of the µC or they may be short-circuit with `MOSI` because the logic probe measured the similar logic levels.

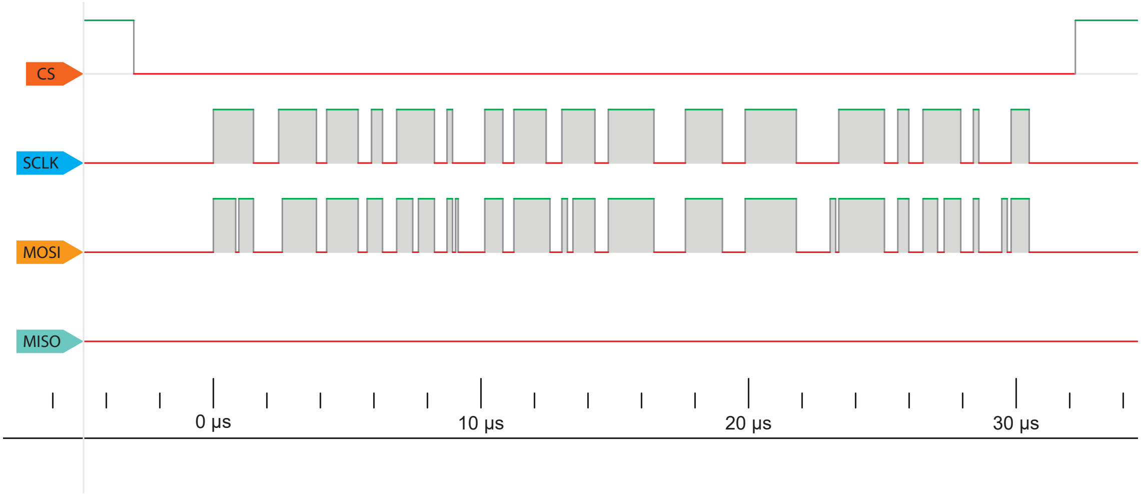


Fig. 6.5: Record of SPI bus communication.

Unfortunately, at the time of the writing this thesis, the MAX31865 sensors were sold out worldwide. Only eight ICs for two devices were ordered as free samples from Maxim Integrated. Unfortunately the second device has a problem with flashing BLE firmware. There still has not been determined where the problem is.

6.5 Balance test

The wireless thermometer will measure the temperature on a real motor. The Figure 6.7 captures the measured motor.

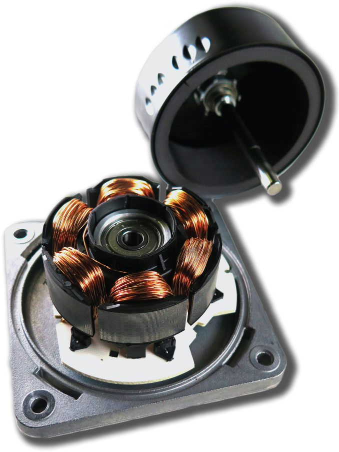


Fig. 6.6: Tested motor.

The tested motor with prototype of the wireless thermometer was tested on the test bench. The speed of the motor was increased by 200 RPM up to 7000 RPM. At the speed of 7000 RPM the vibration increased above the permissible limit, so the motor was shut down immediately to prevent damage to the wireless thermometer or the motor. A speed of 6500 RPM was determined to be the recommended limit. The problem during the test could have been caused by manufacturing inaccuracy.

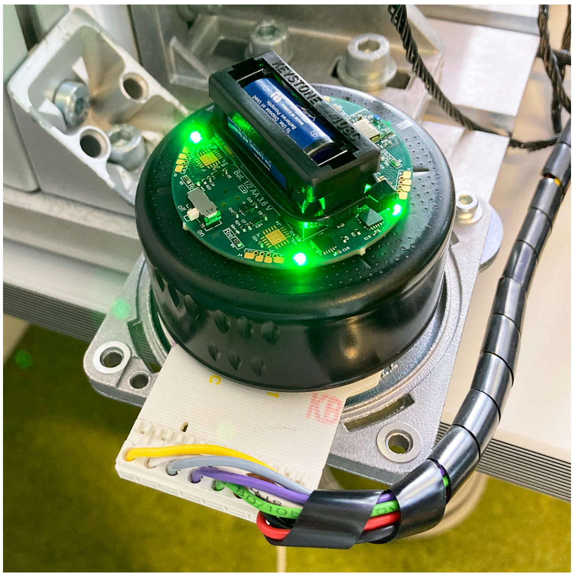


Fig. 6.7: Tested motor.

6.6 Development kit test

Even the developed wireless thermometer failed to run on the custom board, the commissioning can take a place on the development kit[1]. The original firmware was flashed to the development kit. When the device started, the nRF Connect could discover and connect to the wireless thermometer. The SPI communication with four slaves was checked and everything worked well. Also the enable signals (`EN_1`–`EN_4`) for turning on the sensors when the temperature is requested. The measured temperatures were automatically updated in BLE characteristics. The development kit is captured in the Figure 6.8.

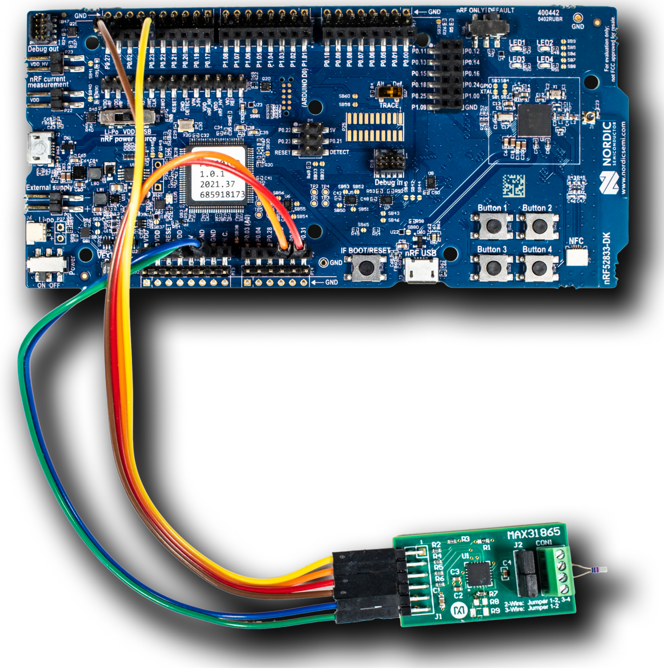


Fig. 6.8: Development kit with sensor.

The design was tested for 24 hours in Linz. It Measured the room and outside temperatures during one day. The main goal of this test was to checked the firmware stability and also this test checked the stability of the developed Python application *BLE Temperature Connect* (which was described in the subsection 5.6.2 on the page 71). The test was successful, the device was able to measure and send the data via BLE for all the time. The Python program recorded all values into CSV file. The first few lines from CSV file is in the Listing 6.1.

[1] Used development kit is nRF52833 DK from Nordic Semiconductor. [27]

Listing 6.1: The measured temperature data from 24-hour test in Linz.

```
time          T1,        T2,        T3,               T4
17:37:22,     16.500,    24.922,    3735928.559,      3735928.559
17:37:23,     16.500,    24.922,    3735928.559,      3735928.559
17:37:23,     16.500,    24.922,    3735928.559,      3735928.559
17:37:24,     16.468,    24.922,    3735928.559,      3735928.559
17:37:24,     16.468,    24.922,    3735928.559,      3735928.559
17:37:25,     16.500,    24.888,    3735928.559,      3735928.559
17:37:26,     16.531,    24.956,    3735928.559,      3735928.559
17:37:28,     16.657,    24.922,    3735928.559,      3735928.559
17:37:30,     16.688,    24.888,    3735928.559,      3735928.559
17:37:31,     16.719,    24.922,    3735928.559,      3735928.559
```

The following Figure 6.9 shows the measured temperature data from Linz. The data were measured by using two identical MAX31865 modules and PT100 sensor in tolerance class B. The first sensor measured the inside temperature and it was connected by 2-wire configuration. The second sensor measured the outside temperature and it was connected by 4-wire configuration because the cable length already affected the accuracy of the measurement.

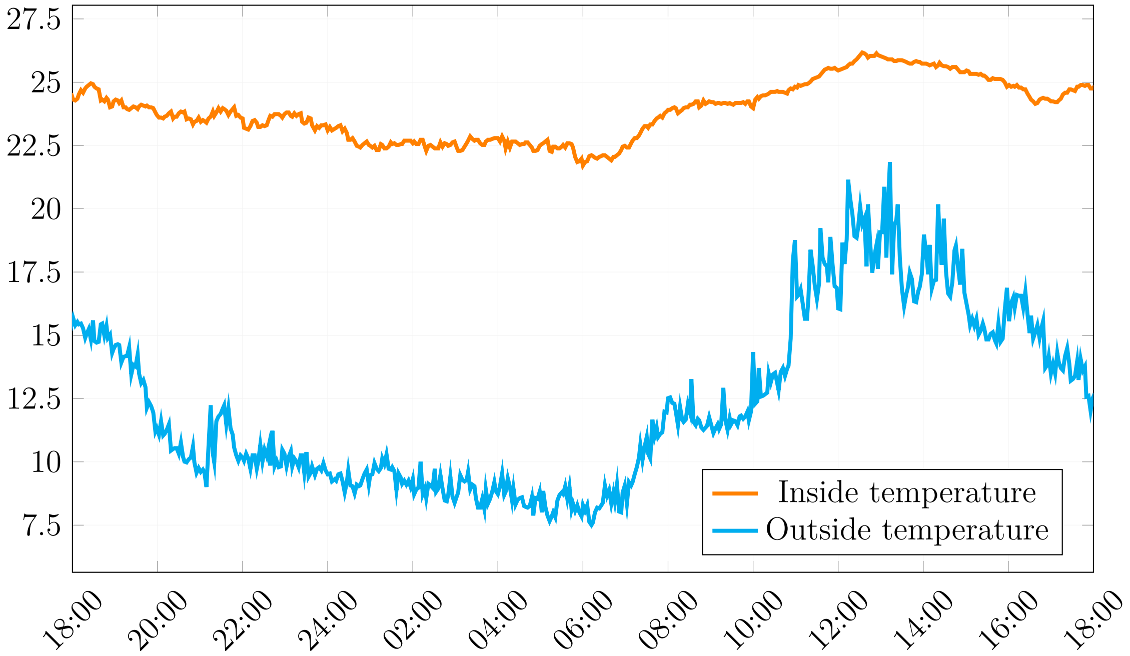


Fig. 6.9: Measured temperature data between 25th to 26th April in Linz.

6.7 Accuracy measurement

Even though the SPI bus could not start running, the accuracy of the MAX31865 was measured using the original module from Maxim Integrated [28]. This module is connected exactly the same as the MAX317865 on the designated wireless thermometer, so we can state that the accuracy measurement will be the same. The control measurement was performed using a universal laboratory electric oven UF30 from Memmert company. The PT100 class A sensor [29] was used as a reference sensor and a precision multimeter with 4-wire resistance measurement was used to measure the temperature. The multimeter automatically converted the measured resistance of the PT100 to temperature so that the temperature could be read directly on the display. A K-type thermocouple was used as an additional control measurement.

Tab. 6.1: Calibration table.

Measured PT100	Reference PT100	Thermocouple	Calibration	Calibration
[°C]	[°C]	[°C]	[°C]	[%]
21.68	22.47	22.00	+0.80	3.54
30.54	31.27	30.80	+0.72	2.32
31.05	31.84	31.40	+0.79	2.48
40.04	40.72	40.50	+0.68	1.67
50.90	51.40	50.70	+0.50	0.97
59.23	60.02	59.80	+0.79	1.32
71.45	72.28	72.00	+0.83	1.15
80.00	80.80	80.90	+0.80	0.99
90.30	91.00	91.20	+0.70	0.77
101.50	102.30	102.90	+0.80	0.78
110.10	110.90	111.70	+0.80	0.72
121.70	122.50	123.20	+0.80	0.65
119.00	120.00	120.90	+1.00	0.83
131.78	132.50	133.70	+0.72	0.54
131.09	131.96	133.10	+0.87	0.66
139.30	140.20	141.90	+0.90	0.64
149.50	150.50	151.10	+1.00	0.66
199.03	199.90	202.40	+0.87	0.44

The following Figure 6.10 shows the dependence of the relative calibration deviation δ on the temperature ϑ. The absolute deviation is almost constant in the whole temperature rang. Therefore, you can notice that the relative deviation for high temperatures is lower than for low temperatures.

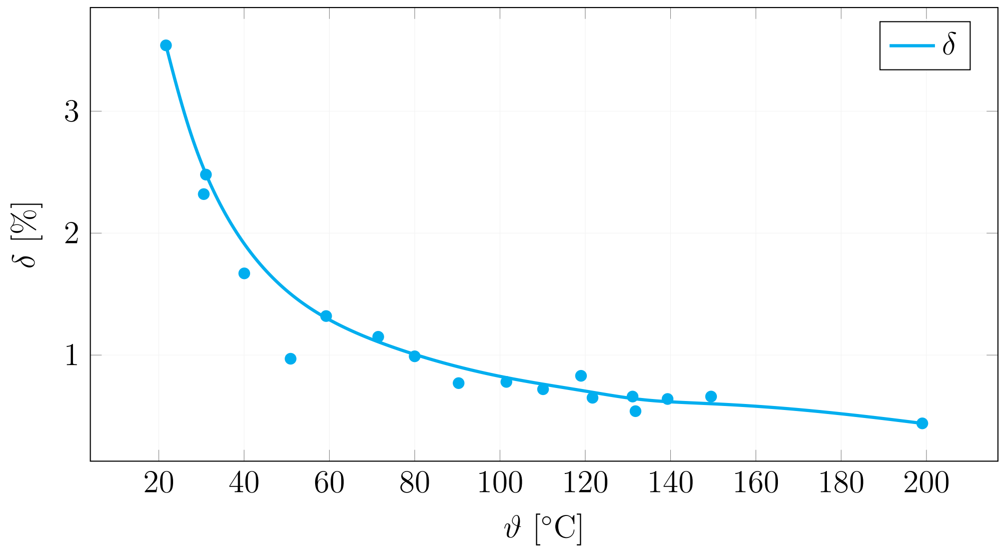


Fig. 6.10: Dependence of the relative calibration deviation δ on the temperature ϑ.

Conclusion

The aim of this Master's thesis was design and realise the wireless thermometer which can measure temperatures in a rotor of an electric motor and can communicate using Bluetooth Low Energy technology.

The realisation is divided into hardware and software parts. In the hardware part, the main task was selecting a suitable microcontroller, measuring integrated circuits, and optimising power consumption. The nRF52833 in the BLE module BL653µ has been selected as the main microcontroller. The main advantage of this µC is that it is a system on the chip, so it can provide the Bluetooth Low Energy communication itself, which makes the implementation easier. The MAX31865 chip from Maxim Integrated has been selected as the main temperature measuring integrated circuit. It has been chosen because of its accuracy and simplicity of communication. The shape of printed circuit board is a circle of 50 mm diameter.

In the software part it was necessary to solve the BLE implementation and to code a program that would measure the temperatures and update the values over the BLE profile. The complete firmware has been written in C language. The communication between the µC and the temperature sensors is performed using SPI bus, which has been implemented using examples from Nordic's SDK and available libraries from Adafruits. To implement BLE communication, an example from the microcontroller manufacturer has been used, which has been supplemented with the necessary profile for four-sensor temperature measurement.

The designated wireless thermometer has been assembly and the firmware was flashed into the microcontroller. Unfortunately, the functionality of the device could not be verified due to a faulty technological process. There was a bad contact on the SPI bus on a function sample and due to the manufacturers' long lead times and the fault could not be resolved. Therefore, the design of the wireless thermometer has been verified on a development kit with original MAX31865. The accuracy of the measurement is ±0.8°C in the range of +20 to +200°C. Also the balance test was passed and the wireless thermometer can rotate up to 6500 RPM.

As a continuance of this work, the client application can be improved. Through which you can communicate with the wireless thermometer and set the measurement properties directly. Thus, it will no longer be necessary to connect the thermometer to the computer via cable before each setting.

Bibliography

1. KREIDL, Marcel. *Měření teploty: senzory a měřící obvody*. Praha: BEN - technická literatura, 2005. ISBN 80-730-0145-4.
2. JOSEPH, Wu. *A Basic Guide to RTD Measurements*. Texas Instruments, 2018. Available also from: `https://www.ti.com/lit/an/sbaa275/sbaa275.pdf?ts=1648556675280&ref_url=https%253A%252F%252Fwww.google.com%252F`.
3. *Industrial platinum resistance thermometers and platinum temperature sensors: EN 60751:2008*. Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2014.
4. PIYU, Dhaker. *Introduction to SPI Interface*. Analog Dialogue, 2018. Available also from: `https://www.analog.com/media/en/analog-dialogue/volume-52/number-3/introduction-to-spi-interface.pdf`.
5. MAXIM, Integrated. *MAX31865: RTD-to-Digital Converter* [online]. 2015 [visited on 2022-03-29]. Available from: `https://datasheets.maximintegrated.com/en/ds/MAX31865.pdf`.
6. AFANEH, M. *Intro to Bluetooth Low Energy: The Easiest Way to Learn BLE*. Amazon Digital Services LLC - KDP Print US, 2018. ISBN 9781790198153. Available also from: `https://books.google.at/books?id=0UhjvwEACAAJ`.
7. BLUETOOTH SIG, Inc. *Bluetooth Core Specification v5.2* [online]. 2019 [visited on 2022-03-29]. Available from: `https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726`.
8. BLAŽEK, Vojtěch. *Sporttester s rozhraním Bluetooth LE* [online]. Brno, 2020 [visited on 2022-04-11]. Available from: `https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=209153`. Master's thesis. Brno University of Technology.
9. NORDIC, Semiconductor. *nRF52833: Product brief* [online]. 2021 [visited on 2022-03-29]. Available from: `https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52833-SoC-Product-Brief.pdf?la=en&hash=9F1B7341D0CB046BB32900C8B520B843EBC8DF06`.
10. BLUETOOTH SIG, Inc. *16-bit UUID Numbers Document* [online]. 2022 [visited on 2022-04-07]. Available from: `https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf`.
11. BLUETOOTH SIG, Inc. *Health Thermometer Profile* [online]. 2011 [visited on 2022-05-10]. Available from: `https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=238687`.

12. *Primary batteries – Part 1: General: EN IEC 60086-1 ed.6:2021*. Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2021.

13. TADIRAN, Batteries. *SL-750: LTC Batteries* [online]. 2019 [visited on 2022-04-04]. Available from: `https://tadiranbatteries.de/wp-content/uploads/2021/05/SL-750.pdf`.

14. TADIRAN, Batteries. *SL-550: LTC Batteries* [online]. 2019 [visited on 2022-04-04]. Available from: `https://tadiranbatteries.de/wp-content/uploads/2021/05/SL-550.pdf`.

15. KEYSTONE. *108: Battery holder, 1/2 AA* [online]. 1996 [visited on 2022-04-04]. Available from: `https://www.keyelco.com/product-pdf.cfm?p=922`.

16. KEYSTONE. *108C: Cover, 1/2 AA battery holder* [online]. 1989 [visited on 2022-04-04]. Available from: `https://www.keyelco.com/product-pdf.cfm?p=3961`.

17. STMICROELECTRONICS. *STLQ020: 200 mA ultra-low quiescent current LDO* [online]. 2021 [visited on 2022-04-04]. Available from: `https://www.st.com/resource/en/datasheet/stlq020.pdf`.

18. MICREL, Inc. *MIC94300: 200mA Switch with Ripple Blocker Technology* [online]. 2012 [visited on 2022-04-04]. Available from: `https://ww1.microchip.com/downloads/en/DeviceDoc/MIC94300.pdf`.

19. TEXAS, Instruments. *SN74LVC125A: Quadruple Bus Buffer Gate With 3-State Outputs* [online]. 2015 [visited on 2022-03-29]. Available from: `https://www.ti.com/lit/ds/symlink/sn74lvc125a.pdf?ts=1648471374266&ref_url=https%253A%252F%252Fwww.google.com%252F`.

20. LAIRD. *BL653µ Series: Datasheet v1.9* [online]. 2022 [visited on 2022-03-29]. Available from: `https://www.lairdconnect.com/documentation/datasheet-bl653-micro-series`.

21. NORDIC, Semiconductor. *nRF52833: Product Specification v1.5* [online]. 2021 [visited on 2022-03-29]. Available from: `https://infocenter.nordicsemi.com/pdf/nRF52833_PS_v1.5.pdf`.

22. DVOŘÁK, Leoš. *Soustava hmotných bodů*. MFF UK Praha, 2016. Available also from: `https://kdf.mff.cuni.cz/vyuka/FyzikaI/FyzI_04_SoustavaHmotnychBodu_ver_0.pdf`.

23. INDUSTRIES, Adafruit. *Adafruit MAX31865* [online]. 2020 [visited on 2022-05-15]. Available from: `https://github.com/adafruit/Adafruit_MAX31865`.

24. SARKAR, Shantanu. Platinum RTD sensor based multi-channel high-precision temperature measurement system for temperature range -100°C to +100°C using single quartic function. *Cogent Engineering* [online]. 2018, vol. 5, no. 1, p. 1558687 [visited on 2022-05-17]. Available from DOI: 10.1080/23311916.2018.1558687.

25. SEMICONDUCTOR, Nordic. *nRF Connect for Desktop: Cross-platform development software for Nordic Products* [online]. 2022 [visited on 2022-05-09]. Available from: https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop.

26. BLIDH, Henrik. *bleak Documentation: Release 0.14.3* [online]. 2022 [visited on 2022-05-09]. Available from: https://bleak.readthedocs.io/en/latest/.

27. BLUETOOTH SIG, Inc. *nRF52833 DK: Development kit* [online]. 2020 [visited on 2022-05-10]. Available from: https://www.nordicsemi.com/Products/Development-hardware/nRF52833-DK.

28. MAXIM, Integrated. *MAX31865PMB1 Peripheral Module: Evaluates: MAX31865* [online]. 2014 [visited on 2022-05-17]. Available from: https://datasheets.maximintegrated.com/en/ds/MAX31865PMB1.pdf.

29. SENSORS, TEWA TEMPERATURE. *Pt100: Platinum Temperature Sensor* [online]. 2018 [visited on 2022-05-17]. Available from: https://www.tme.eu/Document/971f62ae79ad8212f4b583042055630b/TT-PT100A-2050-11-AuNi.pdf.

30. DVOŘÁK, Leoš. *Soustava hmotných bodů.* MFF UK Praha, 2016. Available also from: https://kdf.mff.cuni.cz/vyuka/FyzikaI/FyzI_04_SoustavaHmotnychBodu_ver_0.pdf.

31. TE, Connectivity. *Pt100: Platinum Temperature Sensors* [online]. 2020 [visited on 2022-03-29]. Available from: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FPTF-FAMILY%7FA3%7Fpdf%7FEnglish%7FENG_DS_PTF-FAMILY_A3.pdf%7FCAT-RTD0046.

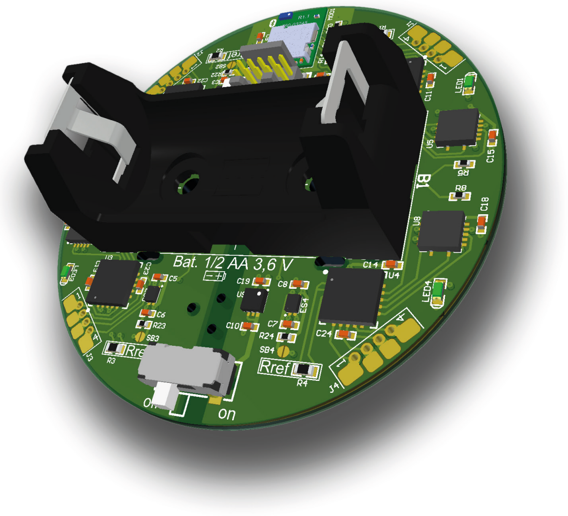
Symbols and abbreviations

Symbol	Unit	Description
α	[K^{-1}]	Temperature coefficient of resistance
ϑ	[°C]	Temperature
A	[K^{-1}]	PT100 sensor material constant
B	[K^{-2}]	PT100 sensor material constant
C	[K^{-4}]	PT100 sensor material constant
I	[A]	Measuring current flowing through RTD
m	[kg]	Weight of mass
R_ϑ	[Ω]	Sensor resistance at temperature ϑ
R_0	[Ω]	Sensor resistance at 0 °C
R_{100}	[Ω]	Sensor resistance at 100 °C
R_m	[Ω]	Measured resistance
R_w1	[Ω]	Resistance of the first wire
R_w2	[Ω]	Resistance of the second wire
R_w3	[Ω]	Resistance of the third wire
R_w4	[Ω]	Resistance of the fourth wire
R_w23	[Ω]	Sum of resistance the second and third wire
R_IN	[Ω]	Input resistance
R_REF	[Ω]	Reference resistor
V_ϑ	[V]	Voltage drop across the RTD at temperature ϑ
V_REF	[V]	Voltage drop across the reference resistor
x_c	[m]	Distance centre of mass from origin
y_c	[m]	Distance centre of mass from origin
ADC		Analog-to-digital converter
AES		Advanced Encryption Standard
ATT		Attribute Protocol
BLE		Bluetooth Low Energy
BR/EDR		Bluetooth Basic Rate/Enhanced Data Rate
BT		Bluetooth
CCM		Counter with cypher block chaining message authentication code
CPHA		Clock Phase
CPOL		Clock Polarity
CPU		Central Processing Unit
CRC		Cyclic Redundancy Code
CS		Chip Select
e.g.		Exempli Gratia (from Latin: For Example)
FHSS		Frequency Hopping Spread Spectrum
GAP		Generic Access Profile

GATT	Generic Attribute Profile
HCI	Host Controller Interface
HW	Hardware
I^2C Bus	Inter-Integrated Circuit Bus
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IoT	internet of Things
ISM	Industrial, Scientific and Medical (band)
L2CAP	Logical Link Control and Adaptation Protocol
LDO	Low-Dropout
LED	Light Emitting Diode
LL	Link Layer
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
µC	microcontroller
PC	Personal Computer
PCB	Printed Circuit Board
PDU	Protocol Data Unit
PHY	Physical Layer
PT100	Platinum sensor $R_0 = 100\ \Omega$
PT1000	Platinum sensor $R_0 = 1000\ \Omega$
RF	Radio Frequency
RPM	Revolutions Per Minute
RS-232	Recommended Standard 232
RTD	Resistance Temperature Detector
SCLK	SPI Clock
SDI	Slave Digital Input
SDIO	Secure Digital Input Output
SDO	Slave Digital Output
SIG	Special Interest Group
SM	Security Manager
SPI	Serial Peripheral Interface
SS	Slave Select
SW	Software
SWD	Serial Wire Debug
TX	Transmit
UART	Universal Asynchronous Receiver-Transmitter
UUID	Universally Unique Identifier

List of appendices

Product brief



Wireless thermometer

Four-channel wireless thermometer based on Bluetooth for laboratory temperature measurements.

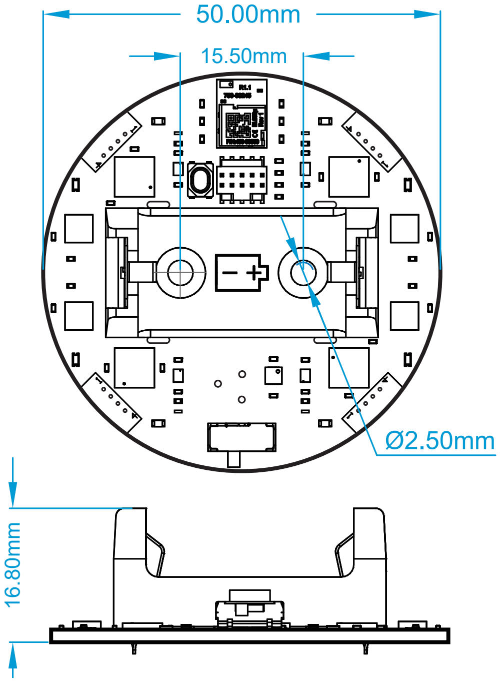
Overview

The wireless thermometer is low-power measuring laboratory equipment. The primary usage is for measuring temperatures in rotating parts of electric machines. However, it can also be used in other applications where the temperatures have to be measured. The key features are low weight, up to four sensors for simultaneous measurement, battery power, long life and the ability to download measurement data directly to a PC or mobile phone in real-time via the standardized Bluetooth Low Energy protocol.

An easy-to-use application is also available to connect the thermometer to a computer and download and save the data to a CSV file. In addition, the thermometer is tested up to 6500 RPM speed which makes it suitable for measuring the temperature of high-speed machines.

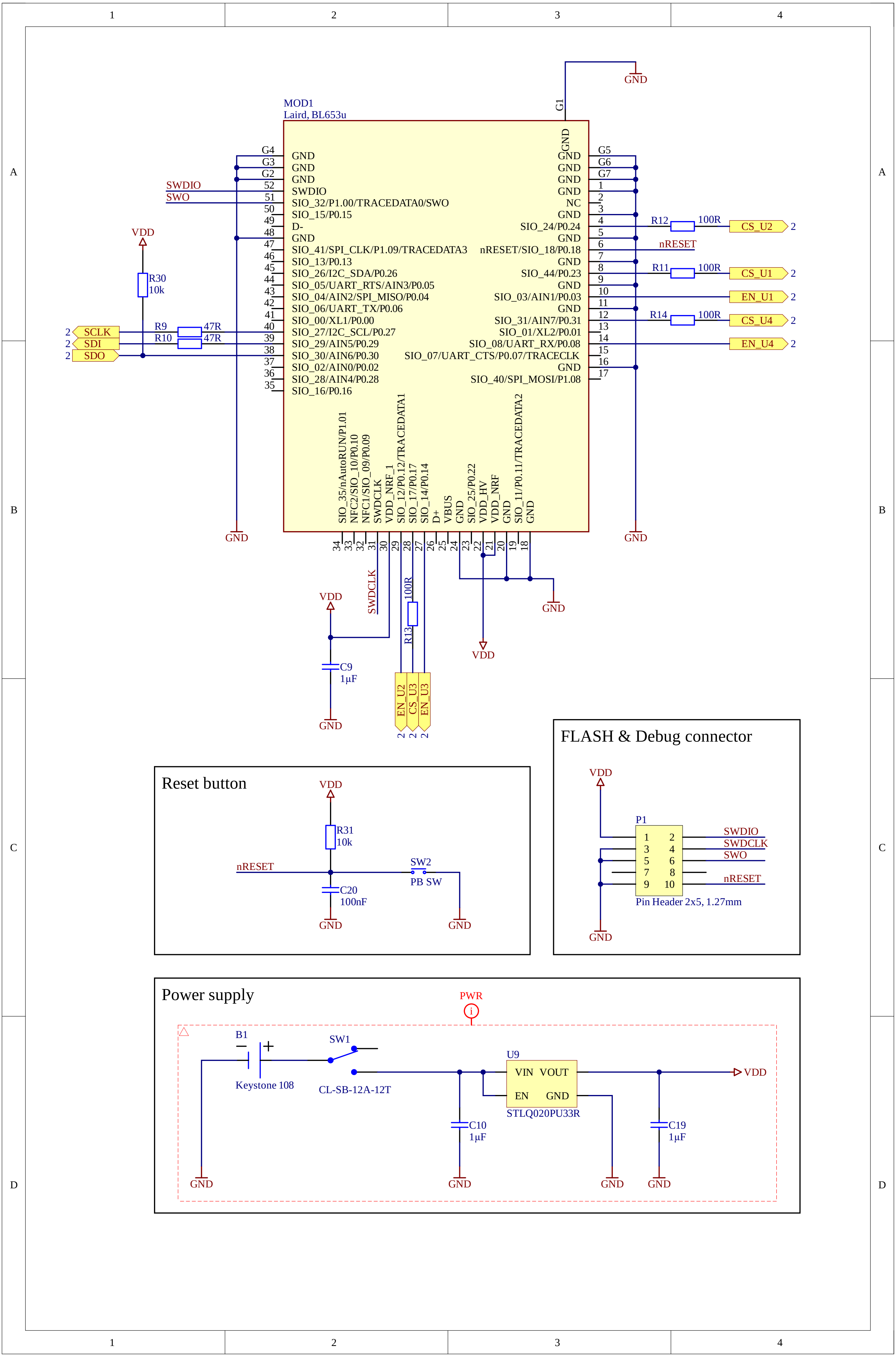
Key features

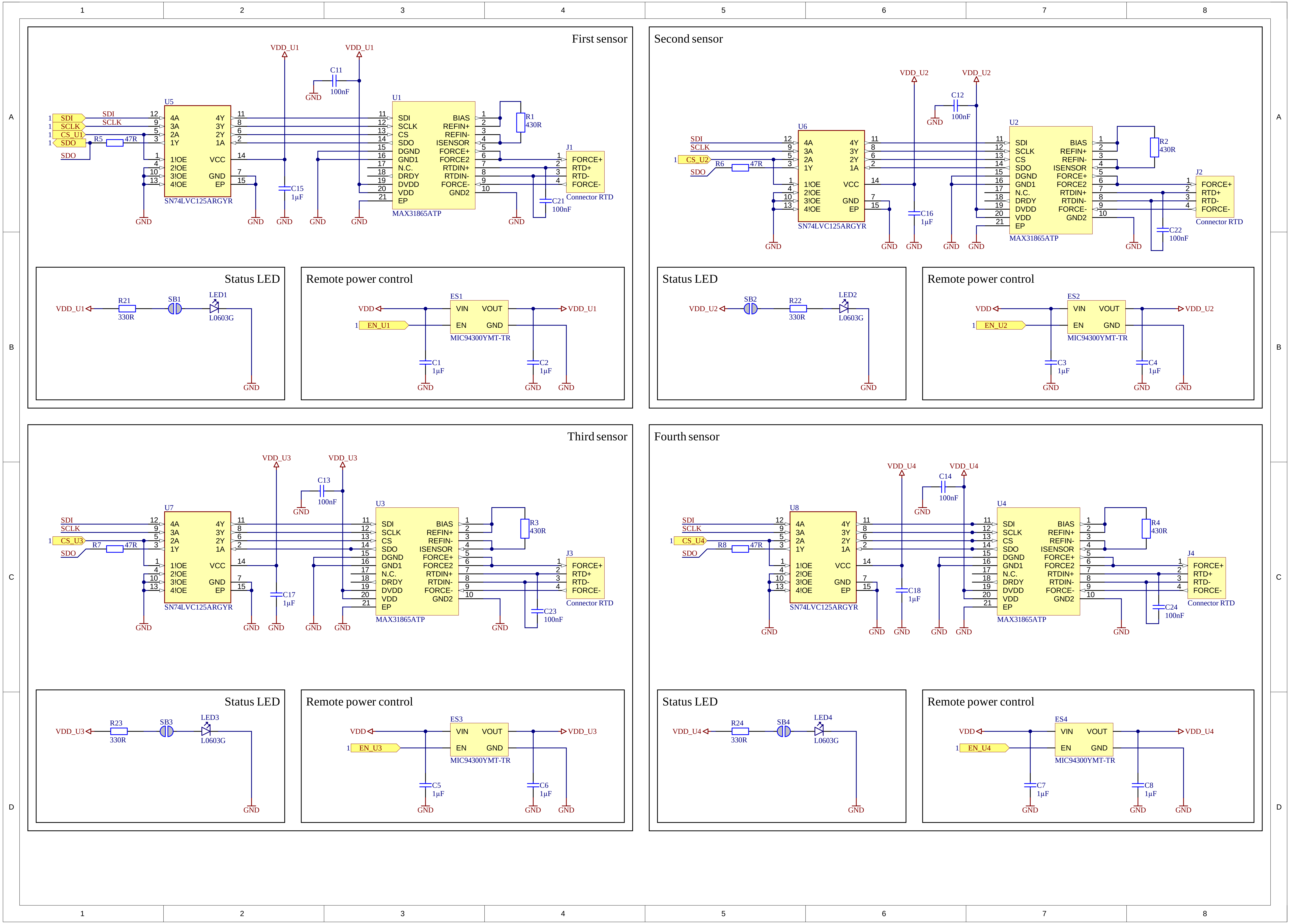
- 4-channel thermometer
  - based on RTD sensors
  - PT100 – PT1000
  - 2, 3 or 4-wire measurement
  - total accuracy of 0.5°C
  - variable measure interval
- Bluetooth
  - range up to 20 m
  - connection with PC or mobile phone
- Battery life
  - up to 1 month (10 s measurement interval)



Specifications

Voltage range:	3.3 to 5.5 V	
Current consumption:	2 mA	(idle state)
	5 mA	(one sensor on)
RTD sensor:	100 to 1000 Ω	(at 0°C)
RTD connection:	2, 3 and 4-wire	
ADC resolution:	15-bit	
Total accuracy:	±0.5°C	
Measurable temp. range:	-200°C to +850°C	
Used battery:	1/2 AA size, voltage 3.6 V recommended: SL350/S	
Max rotating speed:	up to 6500 RPM	
Board temperature range:	-40°C to 105°C	
Weight (without battery):	11.5 g	
(with battery):	20.1 g	

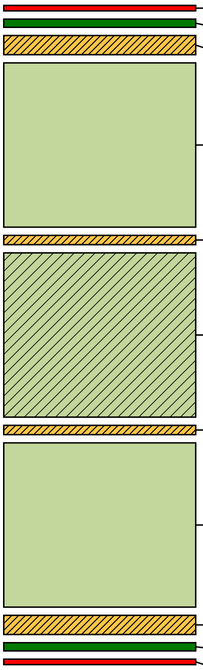

1
2
3
4
A
B
C
D
MOD1
Laird, BL653u
G1
GND
GND
GND
GND
GND
SWDIO
SIO_32/P1.00/TRACEDATA0/SWO
SIO_15/P0.15
D-
GND
SIO_41/SPI_CLK/P1.09/TRACEDATA3
SIO_13/P0.13
SIO_26/I2C_SDA/P0.26
SIO_05/UART_RTS/AIN3/P0.05
SIO_04/AIN2/SPI_MISO/P0.04
SIO_06/UART_TX/P0.06
SIO_00/XL1/P0.00
SIO_27/I2C_SCL/P0.27
SIO_29/AIN5/P0.29
SIO_30/AIN6/P0.30
SIO_02/AIN0/P0.02
SIO_28/AIN4/P0.28
SIO_16/P0.16
G4
G3
G2
52
51
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
GND
GND
GND
GND
NC
GND
SIO_24/P0.24
GND
nRESET/SIO_18/P0.18
GND
SIO_44/P0.23
GND
SIO_03/AIN1/P0.03
GND
SIO_31/AIN7/P0.31
SIO_01/XL2/P0.01
SIO_08/UART_RX/P0.08
SIO_07/UART_CTS/P0.07/TRACECLK
GND
SIO_40/SPI_MOSI/P1.08
G5
G6
G7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
SIO_35/nAutoRUN/P1.01
NFC2/SIO_10/P0.10
NFC1/SIO_09/P0.09
SWDCLK
VDD_NRF_1
SIO_12/P0.12/TRACEDATA1
SIO_17/P0.17
SIO_14/P0.14
D+
VBUS
GND
SIO_25/P0.22
VDD_HV
VDD_NRF
GND
SIO_11/P0.11/TRACEDATA2
GND
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
SWDIO
SWO
VDD
R30
10k
2 SCLK
2 SDI
2 SDO
R9
R10
47R
47R
GND
GND
GND
R12
100R
CS_U2 2
nRESET
R11
100R
CS_U1 2
EN_U1 2
R14
100R
CS_U4 2
EN_U4 2
SWDCLK
VDD
C9
1µF
GND
100R
R13
EN_U2 2
CS_U3 2
EN_U3 2
VDD
GND
Reset button
VDD
R31
10k
nRESET
SW2
PB SW
C20
100nF
GND
GND
FLASH & Debug connector
VDD
P1
1 2
3 4
5 6
7 8
9 10
SWDIO
SWDCLK
SWO
nRESET
Pin Header 2x5, 1.27mm
GND
Power supply
PWR
i
B1
Keystone 108
SW1
CL-SB-12A-12T
U9
VIN VOUT
EN GND
STLQ020PU33R
VDD
C10
1µF
C19
1µF
GND
GND
GND
GND


First sensor
Second sensor
Third sensor
Fourth sensor
Status LED
Remote power control
U1
U2
U3
U4
MAX31865ATP
SDI
SCLK
CS
SDO
DGND
GND1
N.C.
DRDY
DVDD
VDD
EP
BIAS
REFIN+
REFIN-
ISENSOR
FORCE+
FORCE2
RTDIN+
RTDIN-
FORCE-
GND2
U5
U6
U7
U8
SN74LVC125ARGYR
4A
3A
2A
1A
1Y
2Y
3Y
4Y
1!OE
2!OE
3!OE
4!OE
VCC
GND
J1
J2
J3
J4
FORCE+
RTD+
RTD-
FORCE-
Connector RTD
R1 430R
R2 430R
R3 430R
R4 430R
R5 47R
R6 47R
R7 47R
R8 47R
C11 100nF
C12 100nF
C13 100nF
C14 100nF
C15 1µF
C16 1µF
C17 1µF
C18 1µF
C21 100nF
C22 100nF
C23 100nF
C24 100nF
VDD_U1
VDD_U2
VDD_U3
VDD_U4
CS_U1
CS_U2
CS_U3
CS_U4
R21 330R
R22 330R
R23 330R
R24 330R
SB1
SB2
SB3
SB4
LED1 L0603G
LED2 L0603G
LED3 L0603G
LED4 L0603G
ES1
ES2
ES3
ES4
MIC94300YMT-TR
VIN
VOUT
EN
GND
VDD
EN_U1
EN_U2
EN_U3
EN_U4
C1 1µF
C2 1µF
C3 1µF
C4 1µF
C5 1µF
C6 1µF
C7 1µF
C8 1µF
GND
1
2
3
4
5
6
7
8
A
B
C
D

D PCB layers

D.1 Layer stack

The layer stack is set from the used manufacturer.

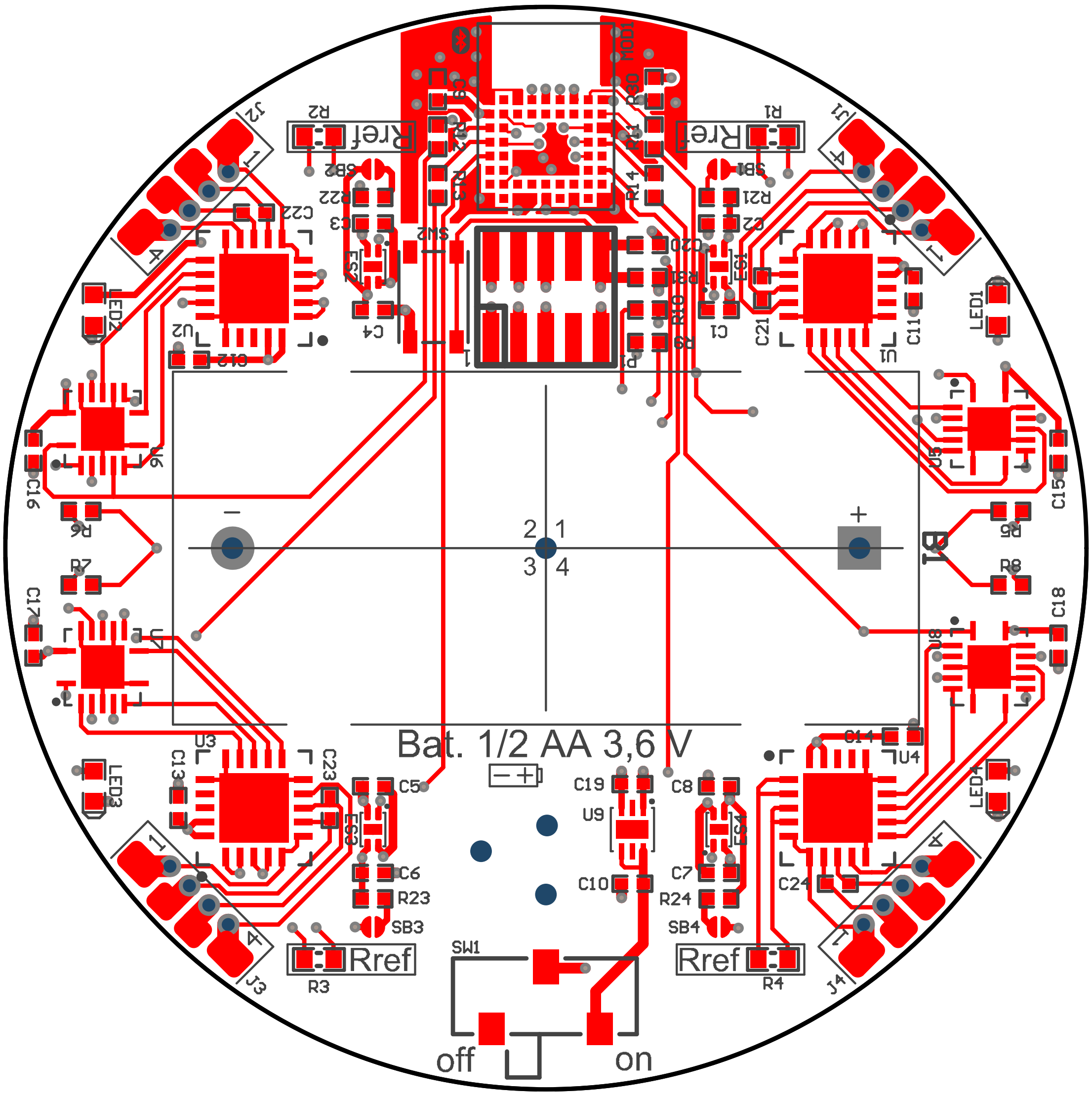


Material	Layer	Thickness	Dielectric Material	Type	Gerber
	Top Overlay			Legend	GTO
Surface Material	Top Solder Mask	0.015mm	Solder Resist	Solder Mask	GTS
Copper	**Top Layer**	**0.035mm**		**Signal**	**GTL**
Prepreg		0.370mm	FR-4	Dielectric	
Copper	**Inner Layer 1 - GND**	**0.017mm**		**Signal**	**G1**
		0.710mm	FR-4	Dielectric	
Copper	**Inner Layer 2 - VDD**	**0.017mm**		**Signal**	**G2**
Prepreg		0.370mm	FR-4	Dielectric	
Copper	**Bottom Layer**	**0.035mm**		**Signal**	**GBL**
Surface Material	Bottom Solder Mask	0.015mm	Solder Resist	Solder Mask	GBS
	Bottom Overlay			Legend	GBO

Total thickness: 1.584mm

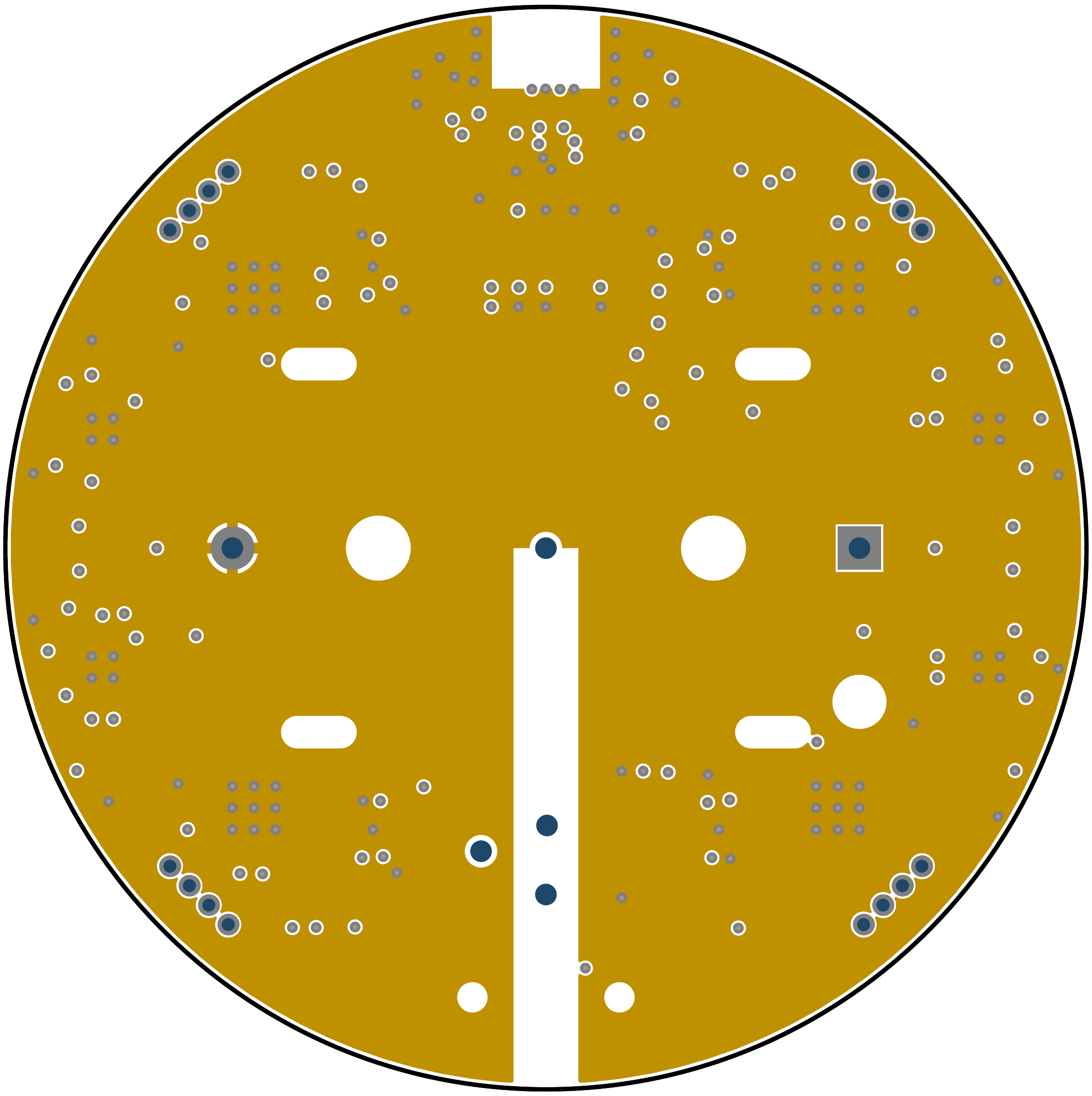
D.2 Top layer and top overlay

Top view



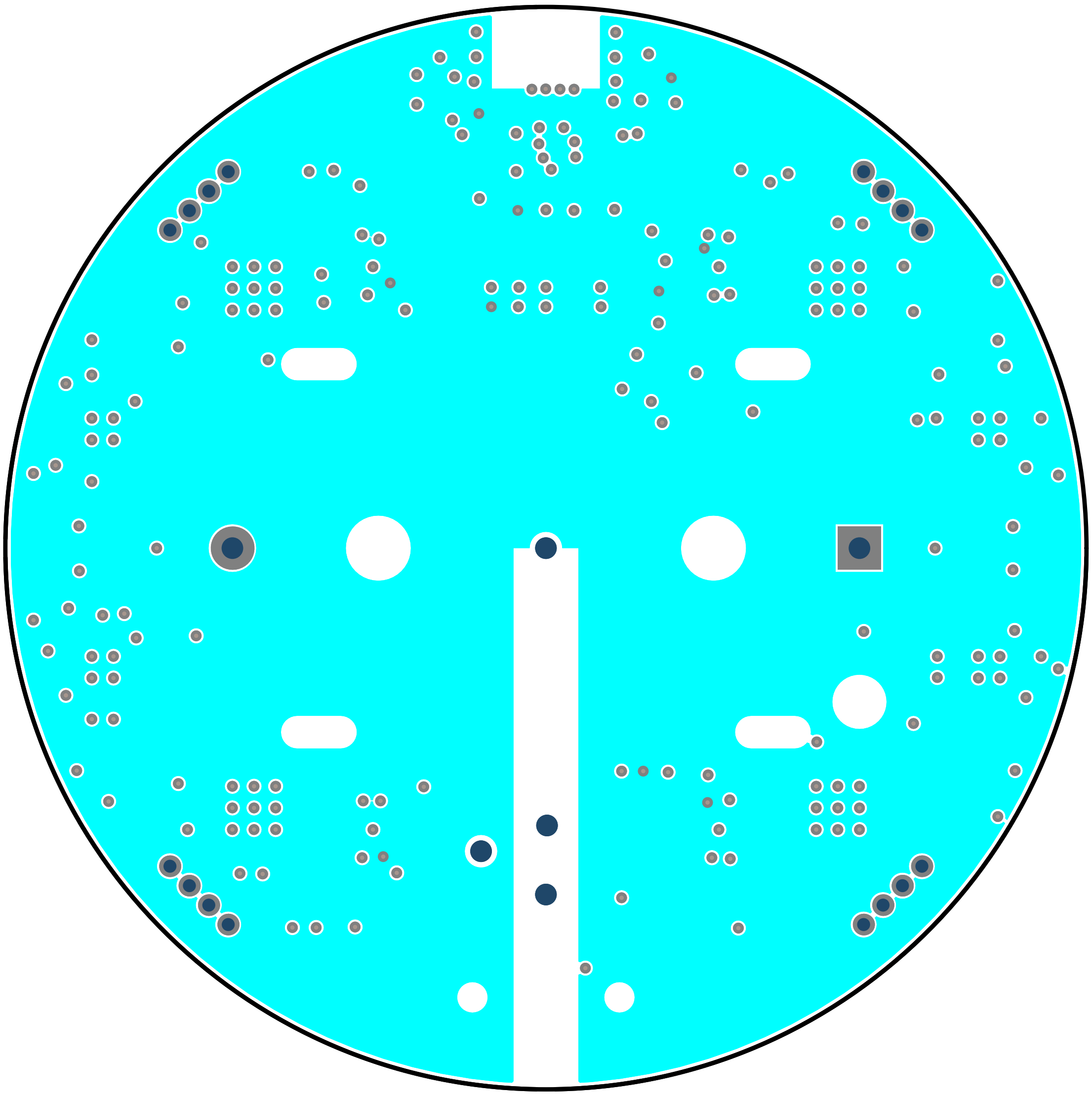
D.3 GND layer – inside layer 1

Top view



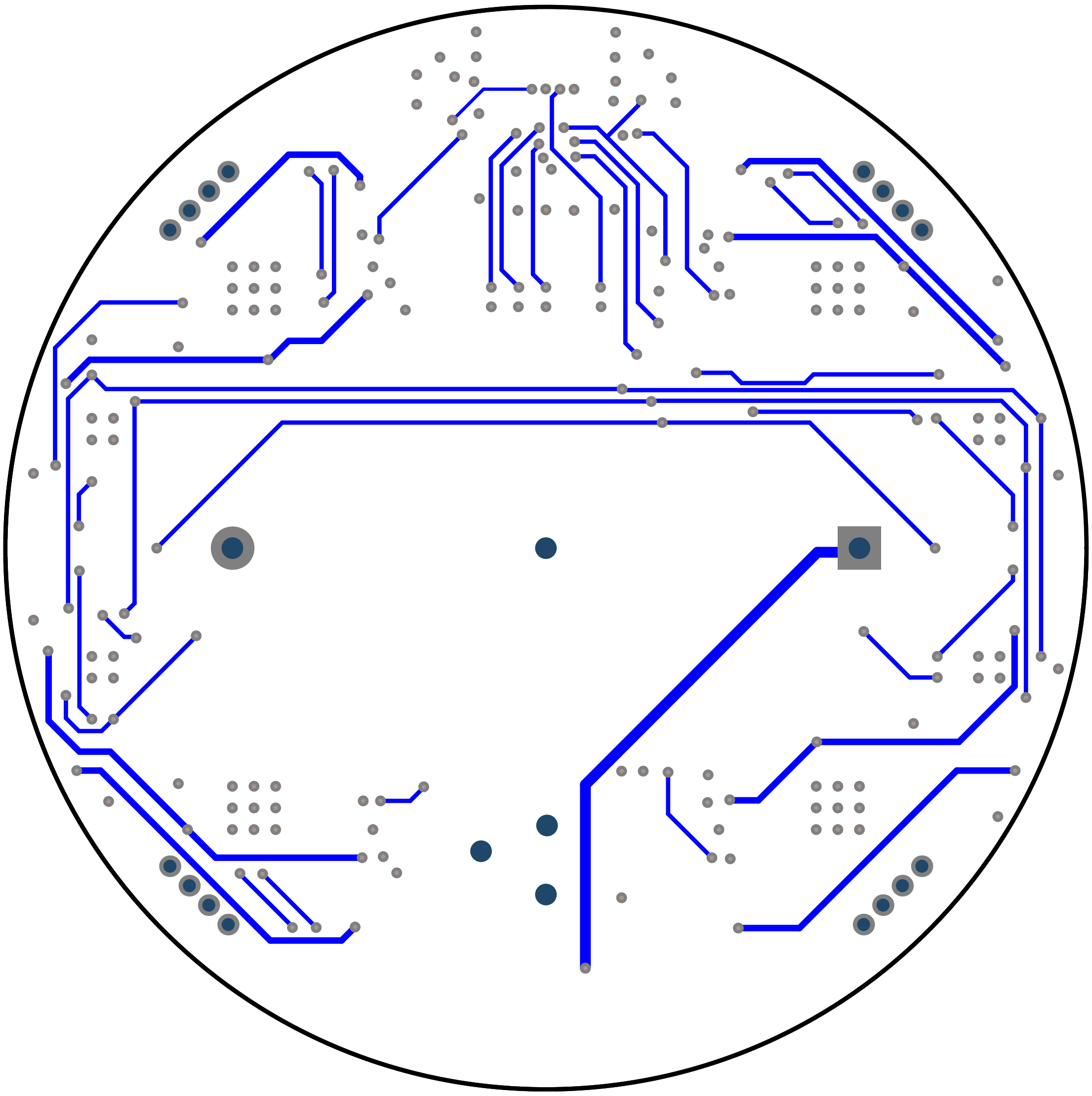
D.4 VDD layer – inside layer 2

Top view



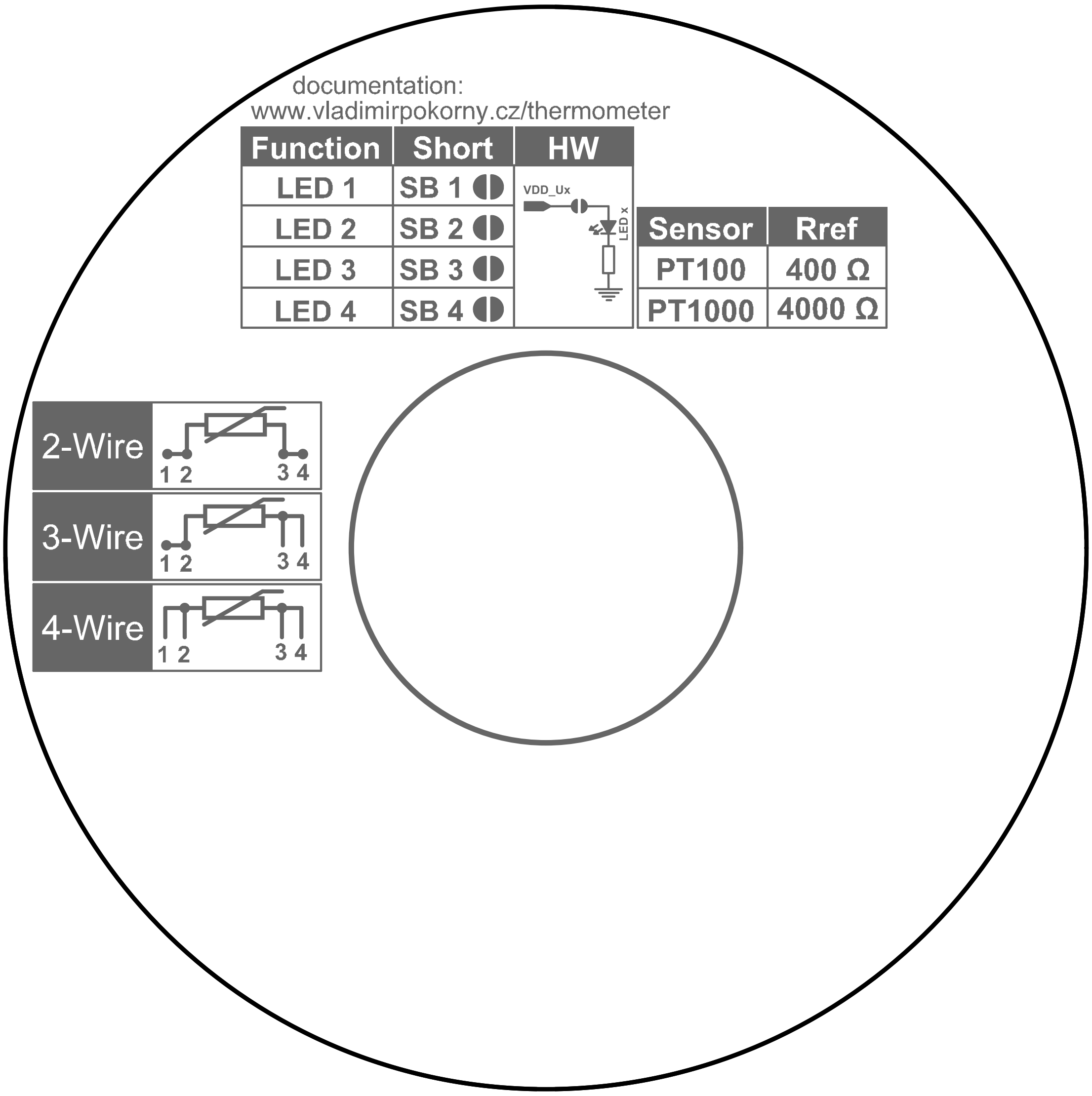
D.5 Bottom layer

Top view



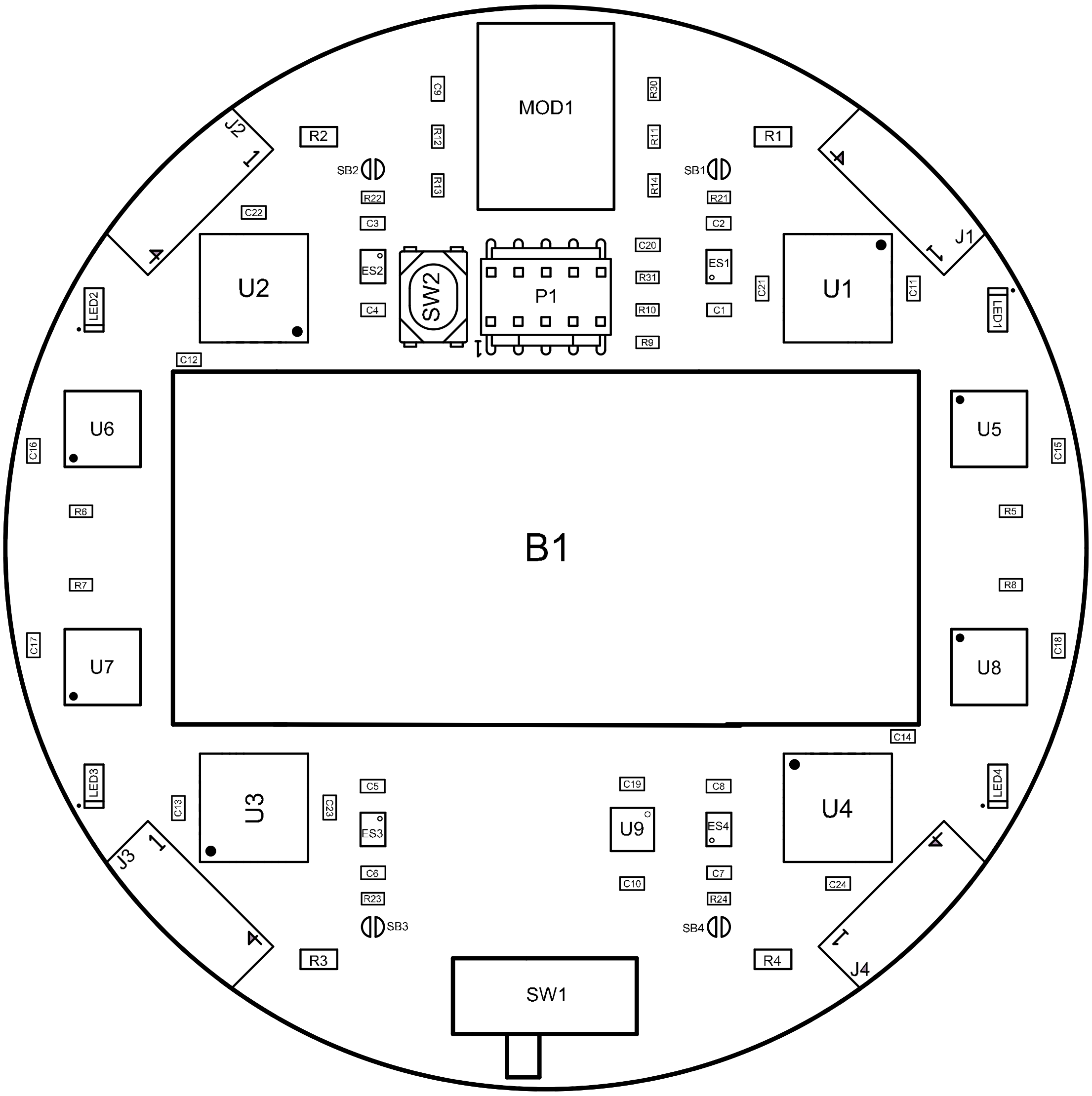
D.6 Bottom overlay – mirrored view

Bottom view



E Assembly

Top view



E.1 List of components

Tab. E.1: List of components.

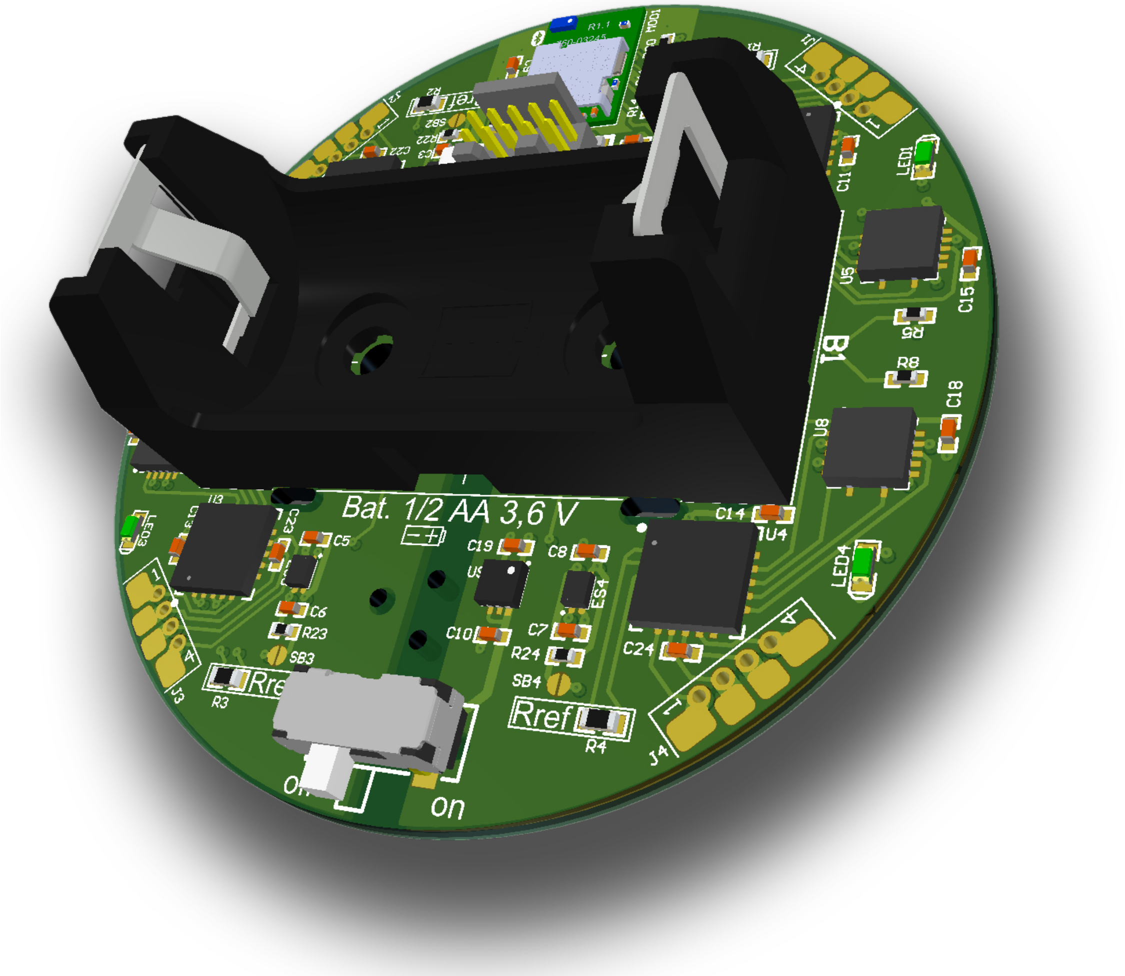
Designator	Value	Package	Note
B1	–	Keystone 108	Battery holder [15]
B2	–	Keystone 108C	Battery holder cover [16]
C1 – C10	1 µF	C1005	Capacitor X7R ±10 %
C11 – C14	100 nF	C1005	Capacitor X7R ±10 %
C15 – C19	1 µF	C1005	Capacitor X7R ±10 %
C20 – C24	100 nF	C1005	Capacitor X7R ±10 %
ES1 – ES4	MIC94300YMT-TR	MLF-4	Power load switch [18]
LED1 – LED4	Green (2.65 V & 2 mA)	LED1608	Green LED
MOD1	Laird BL653µ	–	Bluetooth module [20]
P1	10-pin header	1.27 mm pin pitch	SWD flash connector
R1 – R4	430 Ω	R1005	Resistor ±0.1 %
R5 – R10	47 Ω	R1005	Resistor ±5 %
R11 – R14	100 Ω	R1005	Resistor ±5 %
R21 – R24	330 Ω	R1005	Resistor ±5 %
R30 – R31	10 kΩ	R1005	Resistor ±5 %
SW1	CL-SB-12A-12T	–	Main switch
SW2	PTS810SJK250SMTRLFS	–	Reset button
U1 – U4	MAX31865ATP	TQFN-20	MAX31865 [5]
U5 – U8	SN74LVC125ARGYR	VQFN-14	Quadruple bus buffer [19]
U9	STLQ020PU33R	DFN-6	3.3 V low-dropout voltage regulator [17]

E.2 Additional components

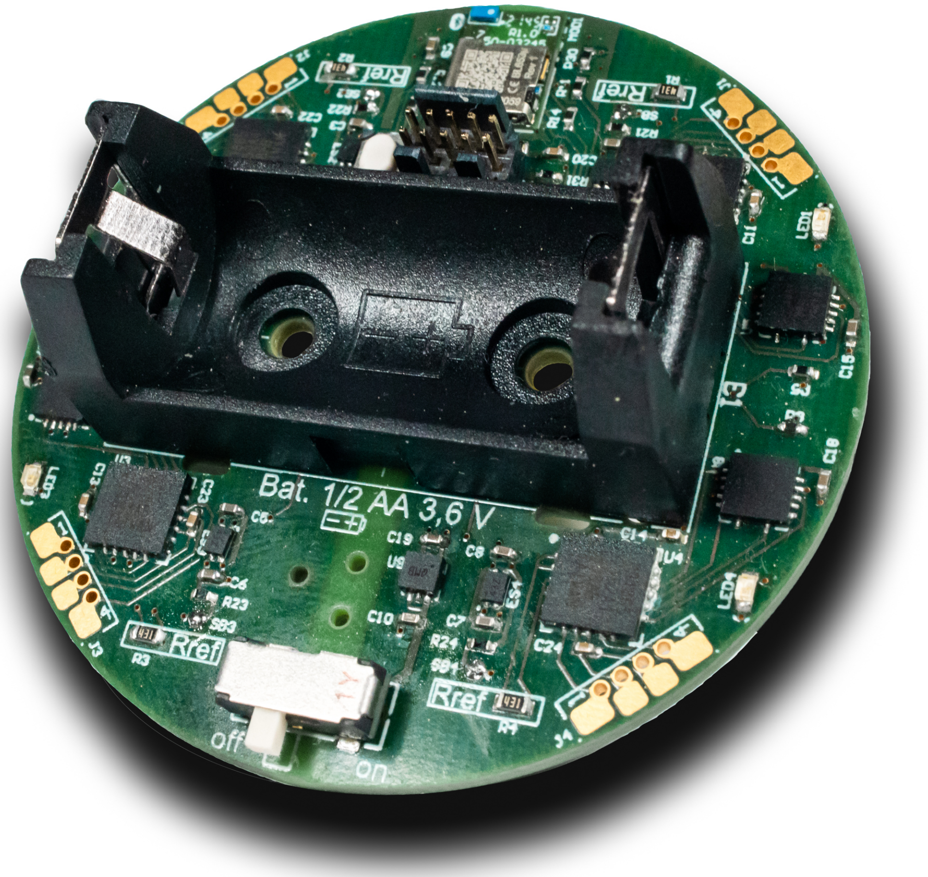
Tab. E.2: List of additional components.

Name	Value	Note
Battery	SL-350/S	1/2 AA size; 3.6 V; 1200 mAh
Cable tie	–	width: 2 mm; long: 100 mm; thick: 1 mm
RTD sensor	PT100 – PT1000	$I_{\mathrm{RTD\_MAX}} = 5.75$ mA
Flash cable	10-pin SWD cable	pin pitch 1.27 mm
J-link programmer		programmer for ARM µC with JTAG

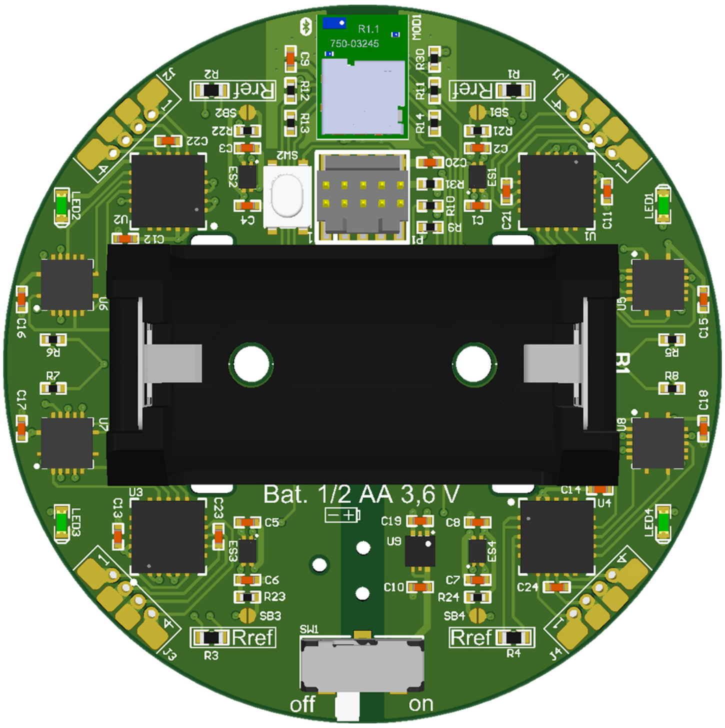
F 3D render


Bat. 1/2 AA 3,6 V
B1
on
Rref
R4
Rre
R3
J3
J4
LED1
LED3
LED4
U3
U4
U5
U8
ES4
C5
C6
C7
C8
C10
C11
C14
C15
C18
C19
C23
C24
R5
R8
R23
R24
SB3
SB4
MOD1

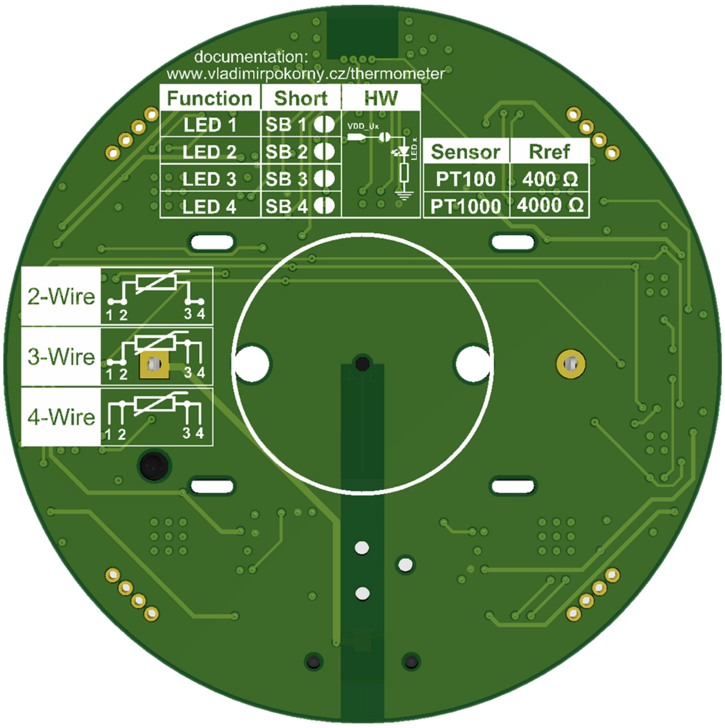
G Real product



G.1 3D Top view



G.2 3D Bottom view



H Components position

Pick & Place data table

Tab. H.1: Components positions and weights.

Designator	Part name	m	x	y	r	Quadrant
		[mg]	[mm]	[mm]	[mm]	
MOD1	Laird, BL653u	940,00	0,00	20,00	20,00	1.
U1	MAX31865ATP	64,00	13,50	12,00	18,06	1.
U2	MAX31865ATP	64,00	-13,50	12,00	18,06	2.
U3	MAX31865ATP	64,00	-13,50	-12,00	18,06	3.
U4	MAX31865ATP	64,00	13,50	-12,00	18,06	4.
U5	SN74LVC125ARGYR	32,00	20,50	5,50	21,22	1.
U6	SN74LVC125ARGYR	32,00	-20,50	5,50	21,22	2.
U7	SN74LVC125ARGYR	32,00	-20,50	-5,50	21,22	3.
U8	SN74LVC125ARGYR	32,00	20,50	-5,50	21,22	4.
U9	STLQ020PU33R	12,00	4,00	-13,00	13,60	4.
ES1	MIC94300YMT-TR	2,40	8,00	13,00	15,26	1.
ES2	MIC94300YMT-TR	2,40	-8,00	13,00	15,26	2.
ES3	MIC94300YMT-TR	2,40	-8,00	-13,00	15,26	3.
ES4	MIC94300YMT-TR	2,40	8,00	-13,00	15,26	4.
B1	Battery Holder 108	4300,00	0,00	0,00	0,00	3.
SW1	CL-SB-12A-12T	210,00	0,00	-20,75	20,75	3.
SW2	PTS810SJK250SMTRLFS	191,00	-5,20	11,60	12,71	2.
C1	1µF	1,50	8,00	11,00	13,60	1.
C2	1µF	1,50	8,00	15,00	17,00	1.
C3	1µF	1,50	-8,00	15,00	17,00	2.
C4	1µF	1,50	-8,00	11,00	13,60	2.
C5	1µF	1,50	-8,00	-11,00	13,60	3.
C6	1µF	1,50	-8,00	-15,00	17,00	3.
C7	1µF	1,50	8,00	-15,00	17,00	4.
C8	1µF	1,50	8,00	-11,00	13,60	4.
C9	1µF	1,50	-5,00	21,20	21,78	2.
C10	1µF	1,50	4,00	-15,50	16,01	4.
C11	1nF	1,50	17,00	12,00	20,81	1.
C12	1nF	1,50	-16,50	8,70	18,65	2.
C13	1nF	1,50	-17,00	-12,00	20,81	3.
C14	1nF	1,50	16,50	-8,70	18,65	4.
C15	1µF	1,50	23,70	4,50	24,12	1.
C16	1µF	1,50	-23,70	4,50	24,12	2.
C17	1µF	1,50	-23,70	-4,50	24,12	3.
C18	1µF	1,50	23,70	-4,50	24,12	4.
C19	1µF	1,50	4,00	-10,90	11,61	4.
C20	1nF	1,50	4,70	14,00	14,77	1.
C21	1nF	1,50	10,00	12,00	15,62	1.
C22	1nF	1,50	-13,50	15,50	20,55	2.
C23	1nF	1,50	-10,00	-12,00	15,62	3.
C24	1nF	1,50	13,50	-15,50	20,55	4.
P1	Pin Header 2x5, 1.27mm	800,00	0,00	11,60	11,60	1.
R1	430R	2,00	10,50	19,00	21,71	1.
R3	430R	2,00	-10,50	-19,00	21,71	3.
R2	430R	2,00	-10,50	19,00	21,71	2.
R4	430R	2,00	10,50	-19,00	21,71	4.
R5	47R	0,65	21,50	1,70	21,57	1.
R6	47R	0,65	-21,50	-1,70	21,57	3.
R7	47R	0,65	-21,50	-1,70	21,57	3.

Tab. H.1: Components positions and weights (continued).

Designator	**Part name**	m	x	y	r	**Quadrant**
		[mg]	[mm]	[mm]	[mm]	
R8	47R	0,65	21,50	-1,70	21,57	4.
R9	47R	0,65	4,70	9,50	10,60	1.
R10	47R	0,65	4,70	11,00	11,96	1.
R11	100R	0,17	5,00	19,00	19,65	1.
R12	100R	0,17	-5,00	19,00	19,65	2.
R13	100R	0,17	-5,00	16,75	17,48	2.
R14	100R	0,17	5,00	16,75	17,48	1.
R21	330R	0,80	8,00	16,20	18,07	1.
R22	330R	0,80	-8,00	16,20	18,07	2.
R23	330R	0,80	-8,00	-16,20	18,07	3.
R24	330R	0,80	8,00	-16,20	18,07	4.
R30	10k	0,65	5,00	21,20	21,78	1.
R31	10k	0,65	4,70	12,50	13,35	1.
LED1	GREEN	1,81	20,90	11,00	23,62	1.
LED2	GREEN	1,81	-20,90	11,00	23,62	2.
LED3	GREEN	1,81	-20,90	-11,00	23,62	3.
LED4	GREEN	1,81	20,90	-11,00	23,62	4.
Ballast		6919,94	0,13	-4,07	4,08	4.

I Files in the attached CD and Git repository

```
/ .................................................. Root of the attached archive
├── tree_structure.txt ............................. File with folder structure
├── Hardware_Files ................................. Main folder with HW files
│   ├── Altium Designer files
│   │   ├── Wireless_thermometer_BLE_module.SchDoc
│   │   ├── Wireless_thermomometer.PrjPcb .......... Altium Designer project file
│   │   ├── Wireless_thermometer_one_PCB.PcbDoc
│   │   ├── Wireless_thermometer_sensors.SchDoc
│   │   └── Library ................................ Altium Designer components libraries
│   │       ├── Wireless_thermometer.PcbLib
│   │       └── Wireless_thermometer.SchLib
│   ├── Production files ........................... Production folder
│   │   ├── Wireless_thermometer.step
│   │   ├── Gerber ................................. Contains all production Gerber data
│   │   └── Pick Place
│   │       └── Pick_place_wireless_thermometer_one_PCB.txt
│   └── Schematic Layout PDF files ................. Contains all PDF files
│       ├── Assembly.pdf
│       ├── PCB_layer_stack.pdf
│       ├── Schematic_and_PCB_layer.pdf
│       └── 3D render
│           ├── PCB_3D_Print.png
│           └── Wireless_Thermometer_3D-PDF.pdf .... 3D model in PDF file
└── Software files ................................. Main folder with SW files
    ├── BLE Connect app ............................ Client Python application folder
    │   ├── BLE_Connect.py
    │   └── ScanDiscover.py
    └── Firmware ................................... Main folder with firmware files
        ├── read_me_first.txt ...................... Instructions to start development
        └── ble_remote_thermometer
            ├── main.c ............................. Main c-file with the main program
            ├── our_service.c
            ├── our_service.h
            ├── MAX31865
            │   ├── MAX31865.c
            │   └── MAX31865.h
            └── pca10100
                └── s140
                    ├── config
                    │   └── sdk_config.h
                    └── ses
                        ├── ble_remote_thermometer.emProject .... Main SEGGER project
                        ├── ble_remote_thermometer.emSession
                        ├── flash_placement.xml
                        └── Output ......................... Folder with output files
```

www.vladimirpokorny.cz/thermometer/