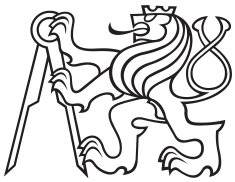
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Control and Bluetooth Communication for Wearable Safety Jacket Applications

Bc. Ladislav Štefka

Supervisor: Ing. Pavel Píša, Ph.D
Field of study: Cybernetics and Robotics
May 2022

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Štefka Ladislav** Personal ID number: **469913**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Control and Bluetooth Communication for Wearable Safety Jacket Applications

Master’s thesis title in Czech:

Ovládání a Bluetooth komunikace pro systémy aktivního osvětlení oděvů

Guidelines:

Safety, visibility and comfort of rescue, emergency and civil services workers as well as of sportsmen operating in low light conditions on plants and highways can be enhanced by active lightning systems. The project focuses on developing active lightning systems firmware and accompanying monitoring and control applications for mobile devices.
1) Familiarize with active lightning system for jackets and Bluetooth technology and discuss selection of appropriate microcontroller for control units and technology for mobile applications
2) Implement firmware for control units which allows local and remote control and provide mechanisms for in application firmware updates.
3) Implement monitoring and control applications for current mainstream mobile platforms.
4) Analyze possibilities to add advanced functions as gesture control, RFID based control etc.
5) Document projects and source code repositories for future development and prepare manuals and scripts for devices bring-up in planned serial production.

Bibliography / sources:

1) PINETIME - An Open Source Smartwatch For Your Favorite Devices. Low Cost, High Fidelity. https://www.pine64.org/pinetime/
2) nRF52833 Product Specification, v1.5, 2020, NORDIC semiconductor, https://infocenter.nordicsemi.com/pdf/nRF52833_PS_v1.5.pdf
3) Townsend, K., Cufí, C., Davidson, R.:Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking 1st Edition, O'Reilly Media; 2014, ISBN: 1491949511

Name and workplace of master’s thesis supervisor:

Ing. Pavel Píša, Ph.D. Department of Control Engineering FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **19.01.2022** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

Ing. Pavel Píša, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Ing. Pavel Píša, Ph.D., for the his continuous support, availability at any possible hour of the day, endless willingness to help, patience, and big portion of motivation during the development and writing of this thesis.

Secondly, I want to thank to Ing. Petr Porazil, designer of the hardware, whose surgically accurate soldering saved me many times and his valuable advice helped significantly.

Not to forget, I would like to thank to Ao.Univ.Prof. Dipl.-Ing. Dr. Thilo Sauter from Vienna University of Technology for his help during my exchange in Vienna.

Special thanks go to my friend and colleague Bc. Petr Knetl who helped me hugely with React Native and development operations on mobile platforms.

Last but not least, I would like to mention my parents for their huge support and SCILIF team for the opportunity to cooperate with them and attend the A+A exhibition.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Signature:

In Prague, May 20, 2022

Abstract

Before the active lighting systems become a standard of personal safety in low-light conditions, it is essential to solve their reliable and practical control. This thesis aims to analyse, design, and implement suitable methods for control of an active lighting system. Eventually, these methods were elicited: remote control by a mobile application and an external floating button, communicating via Bluetooth Low Energy, contactless activation by an RFID tag, and gesture control.

The scope of the thesis comprises a development of a light control unit and floating button firmware, and a mobile application. The light control unit regulates the light guide luminance in predefined light modes and allows the user to change them using the methods described above. The mobile application enables the user to communicate with one or multiple units, remotely change the light mode, read the monitored values, configure the RFID detection and start an OTA firmware upgrade. The external floating button is designed for a fast and comfortable change of a light mode on a paired unit.

The detection of an RFID tag was tested with an external reader. However, later, it was implemented directly in the light control unit by a PWM excitation of the oscillator with synchronous sampling.

Keywords: active lighting system, remote control, IoT, BLE, RFID, react-native, nRF52

Supervisor: Ing. Pavel Píša, Ph.D
Praha, Resslova 9, E-7a

Abstrakt

Předtím než se aktivní systémy osvětlení stanou standardem osobní bezpečnosti v situacých se zhoršenými světelnými podmínkami, je nutné vyřešit, jakým způsobem je jejich uživatelé mohou pohodlně a spolehlivě ovládat. Cílem této práce je analýza, návrh a implementace vhodných způsobů ovládání aktivního systému osvětlení. Pro ovládání byla vybrána mobilní aplikace a externí tlačítko, komunikující standardem Bluetooth Low Energy, bezkontaktní aktivace přiblížením RFID tagu a ovládání gesty.

V rámci práce probíhal vývoj firmware řídící jednotky osvětlení a externího tlačítka a také mobilní aplikace. Řídící jednotka reguluje jas světlovodu v přednastavených světelných módech a umožňuje uživateli jejich změnu pomocí metod popsaných výše. Mobilní aplikace slouží pro komunikaci s jednou nebo více jednotkami, umožňuje vzdálenou změnu světelného módu, čtení monitorovaných veličin, nastavení RFID detekce a OTA upgrade firmwaru. Externí tlačítko je určené pro rychlou a pohodlnou změnu světelnýh módů na spárované jednotce.

Detekce RFID tagu byla otestovaná s využtím zakoupené externí čtečky a poté byla funkce implmentovaná přímo v řídíci jednotce pomocí buzení přes PWM se sychroním vzorkováním nosné.

Klíčová slova: aktivní systém osvětlení, vzdálené ovládání, IoT, BLE, RFID, react-native, nRF52

Contents

1 Introduction 1

2 The Lighting System 3

2.1 Light Control Unit 3

2.2 Floating Button 3

2.3 Light Modes 3

2.3.1 Analysis of Methods of Light Modes Changing 4

2.3.2 Selected Methods of Light Modes Changing 4

2.4 Monitoring & Configuration & FW Upgrade 6

2.5 Controlling the BLE Operation . . 7

2.6 Non-Functional Requirements . . . 7

3 Bluetooth Low Energy & RFID 9

3.1 BLE 9

3.1.1 Physical Layer (PHY) 10

3.1.2 Link Layer (LL) 11

3.1.3 GAP 13

3.1.4 GATT 15

3.2 RFID 17

3.2.1 RFID Tags 17

3.2.2 RFID Readers 17

3.2.3 RFID Frequency Bands 18

3.2.4 RFID Data Transmission . . . 18

4 HW Specification 21

4.1 HW of Light Control Unit 21

4.1.1 Microcontroller (MCU) 22

4.1.2 Radio 24

4.1.3 Battery Measurement Circuit 25

4.1.4 Constant Current LED Driver with SEPIC 27

4.1.5 Battery Charger Circuit 29

4.1.6 Gyroscope/Accelerometer . . . 29

4.1.7 The Overall Design 29

4.2 HW of Floating Button 30

4.2.1 Accelerometer/Magnetometer 30

4.3 HW of RFID Reader and Tag . . 31

4.3.1 RDM6300 RFID Reader 31

4.3.2 Custom Low Power RFID Reader 32

4.3.3 EM4100 RFID Tag 34

5 Software Development - Firmware 35

5.1 Tools Used 35

5.1.1 Programming 35

5.2 nRF52 Memory Layout 36

5.3 Code Structure 37

5.3.1 Configuration Files 38

5.3.2 Application Logic Source Files 38

5.4 BLE Properties and Modes of Operation 38

5.4.1 Advertising and Scanning . . . 39

5.4.2 Connection Parameters 40

5.4.3 Security 40

5.4.4 BLE Services and Characteristics 41

5.5 Application Logic 44

5.5.1 Application States 45

5.5.2 Application Events 46

5.5.3 Application Timer 46

5.6 Light Control Unit 46

5.6.1 BLE Properties 46

5.6.2 User Interface 47

5.6.3 Application Logic 47

5.7 Floating Button 52

5.7.1 BLE Properties 52

5.7.2 User Interface 52

5.7.3 Application Logic 53

5.8 Button Debouncing & Press Detection 56

5.8.1 Button Debouncing 56

5.8.2 Button Application Events . . 56
5.9 Battery, LED and Temperature Measurement 57
5.9.1 Battery Measurement 58
5.9.2 LED Voltage and Current Measurement 58
5.9.3 Sampling 58
5.9.4 Temperature Measurement . . 59
5.9.5 Battery Application Logic . . . 59
5.10 OTA DFU 60
5.10.1 Bootloader 60
5.10.2 Bootup Validation 60
5.10.3 Buttonless DFU Activation . 60
5.10.4 Firmware Upgrade Process . 61
5.10.5 Tools Used 62

6 RFID Tag Detection 63

6.1 EM4100 Protocol Description . . 63
6.2 RDM6300 RFID Reader Data Processing 64
6.3 LC RFID Reader Control and Signal Processing 64
6.3.1 Sampling 65
6.3.2 Signal Processing 65
6.3.3 Detection 66
6.4 RFID Application Logic 67

7 Software Development - Mobile Application 69

7.1 Mobile App Development 69
7.1.1 Native Applications 69
7.1.2 Web Applications 69
7.1.3 Hybrid Applications 70
7.2 Tools Used 70
7.2.1 React Native 70
7.2.2 Expo 71
7.2.3 TestFlight 71
7.3 UI Structure 71
7.3.1 Introduction Screen 71
7.3.2 Devices Screen 71
7.3.3 Device Control Screen 72
7.4 BLE Implementation Details . . . 74
7.4.1 Limitations 76

8 Gesture Control 77

8.1 Inertial Gesture Recognition . . . 77
8.2 Data-set Collecting 77
8.2.1 LSM6DSMTR Setup 78

9 Results 79

9.1 Battery Consumption 79
9.1.1 Static Consumption 79
9.1.2 Consumption of Periodic Events 80
9.1.3 Consumption in Application States 83
9.1.4 Conclusions 83
9.2 Communication Range 86
9.3 Regulation of LED Current 87
9.4 RFID Detection Range 88
9.5 A+A Düsseldorf 88

10 Conclusion 89

A Additional Figures & Schemes 91

B List of Acronyms and Symbols 94

C Bibliography 96

D Sources - Images 99

E Sources - Software 100

Figures

1.1 SunFibre wearable active lighting system 2

3.1 The BLE protocol stack 9
3.2 BLE channels 10
3.3 GFSK modulation 10
3.4 Advertising and scanning 11
3.5 Connection events 12
3.6 Pairing and bonding sequences . 14
3.7 Frequency bands of RFID 18
3.8 Inductively coupled RFID 19

4.1 Block diagram of the light control unit 21
4.2 nRF52833 block diagram 23
4.3 nRF52833 radio block diagram . 25
4.4 Battery measurement circuit . . . 25
4.5 Typical SEPIC topology 27
4.6 DIO5661 functional block diagram 28
4.7 LED driver circuit 28
4.8 Light control unit board 30
4.9 BBC micro:bit.v2 30
4.11 EM4100 RFID tag and RDM6300 RFID reader 31
4.13 RFID resonance circuit 32
4.14 Light control unit with RFID reader and a light guide 33

5.1 nRF52 memory regions 36
5.2 Advertising and scan response packet of light control unit 39
5.3 Section of GATT attribute table - LED Control and Monitor services 42
5.4 Example of communication topology 44
5.5 State diagram - application start (wakeup) 48
5.6 State diagram - connected 49
5.7 State diagram - joining 49
5.8 State diagram - bonding 50
5.9 State diagram - scanning (central) 54
5.10 State diagram - advertising (peripheral) 55
5.11 Button GPIO and timer interrupt handlers (ISRs) 57
5.12 WorkFlow of the DFU process . 61

6.1 EM4100 memory array 63
6.2 Manchester encoding 64
6.3 RDM6300 word 64
6.4 EM4100 RFID signal processing 68

7.1 Introduction and devices screen . 72
7.3 Device control screen and DFU upgrade dialog 73
7.5 Monitoring dialogs 74
7.7 Mobile application component diagram 75

8.1 Examples of recorded gestures . . 78

9.1 Battery consumption - advertising packet 81
9.3 Battery consumption - data packets 82
9.4 Battery consumption - RFID & monitoring 84
9.5 Measured characteristic for input voltage 3.6 V 88

A.1 SCILIF stall at A+A Düsseldorf 91
A.2 The HW design of light control unit board (created by Petr Porazil) 92
A.3 The HW design of light control unit board (created by Petr Porazil) 93

Tables

2.1 Comparison of possible light mode change methods 5

5.1 AD elements of the light control unit 39
5.2 Selected BLE parameters 41
5.3 A list of supported application events 45
5.4 Application states of light control unit 47
5.5 Error states of light control unit. 51
5.6 Application states of the floating button. 53
5.7 Button click application events and their defaul duration 57
5.8 ADC channel parameters 58
5.9 Battery levels and indication colors (on the light control unit) 59

6.1 ADC channel parameters for RFID sampling. 65

8.1 LSM6DSMTR parameters for gesture control 78

9.1 Average current consumption of nRF and application states 80
9.2 Average current consumption of board peripherals 80
9.3 Battery consumption - advertising & connection 81
9.4 Battery consumption - different connection TX power levels 82
9.5 Battery consumption - RFID & monitoring 83
9.6 Battery consumption - system states 85
9.7 Range test results 87
9.8 Range test results 87
9.9 LED regulation results 87

Chapter 1

Introduction

Personal safety in the workplace depends on the own awareness of potential threats and risks. To reduce the risk of injury or harm to health in dark situations or low-light areas, active lighting systems as a new standard in personal safety are emerging [26]. As the overall complexity of such systems increases, it is essential to bear in mind that their control should be as easy and practical as possible and shouldn't disturb its user's work or activity.

The primary concern of this thesis is to design and implement a solution for remote control of an active lighting system. The objective is to develop an embedded control device accompanied by monitoring and control applications for mobile devices.

A comprehensive analysis of possible approaches to this problem and their short comparison is performed in chapter 2. Additionally, the reasons behind selecting particular methods are presented as well. Last but not least, this chapter also contains an introduction to all system parts and the requirements. The next chapter 3 describes the communication technologies, BLE and RFID, selected for the remote control according to the analysis from the previous chapter.
A specification of the selected hardware components, together with reasons behind their selection, is presented in chapter 4. It further provides a theoretical background and an analysis of specific system parts (electrical circuits). Chapter 5 provides a complex view of the firmware development of the lighting system devices (light control unit and floating button). It is the longest chapter containing a lot of details (description of the user interface and the application logic, BLE configurations and an explanation of various software techniques and algorithms used). Nevertheless, a separate chapter 6 is also dedicated to the development of RFID readers. It provides a theoretical background to the EM4100 protocol and the signal processing that was used to obtain tag id from sampled RFID signal. The mobile application development is described in a separate chapter 7 too. It further introduces approaches to mobile application development together with a description of the implemented application. Chapter 8 briefly comments the achievements in recognition of gestures. The last chapter 9 is dedicated to the results, performed tests and measurements.

Motivation & SunFibre Active Lighting System

In order to keep workers safe and visible in low-light conditions, people still rely on reflective tape, a product developed in the 1970s. Tape and other reflective surfaces remain dependent on an external source of light to illuminate them. Due to this dependency, they are very limited in the overall ability to keep workers visible. The active lighting systems address this problem by helping a person to be seen without depending on a secondary illumination source. When a person is visible from all directions, the job site is safer for everyone [26].

One of the pioneers of active lighting systems in the Czech Republic is a company called SCILIF. With many years of experience in this field, they have developed a new patented technology for active lighting systems, which has already proved its commercial viability. **SunFibre Wearable** is an optic fibre lighting system that emits light through optic fibres encased in a textile coating. Side-emitting optic fibres provide visibility in all directions up to 3 kilometers. The properties of the textile coated optic fibres allow easy sewing into products such as jackets or vests, guarantee mechanical durability and washability and do not produce any heat[1].

The personal safety field is rather a conservative environment, and active lighting systems are slowly approaching the markets of traditional reflective tapes. Hence, a hi-tech, remotely controlled active lighting system is rather a niche product. Nevertheless, the motivation for SCILIF company is to demonstrate its technological advantage and broaden its product portfolio.

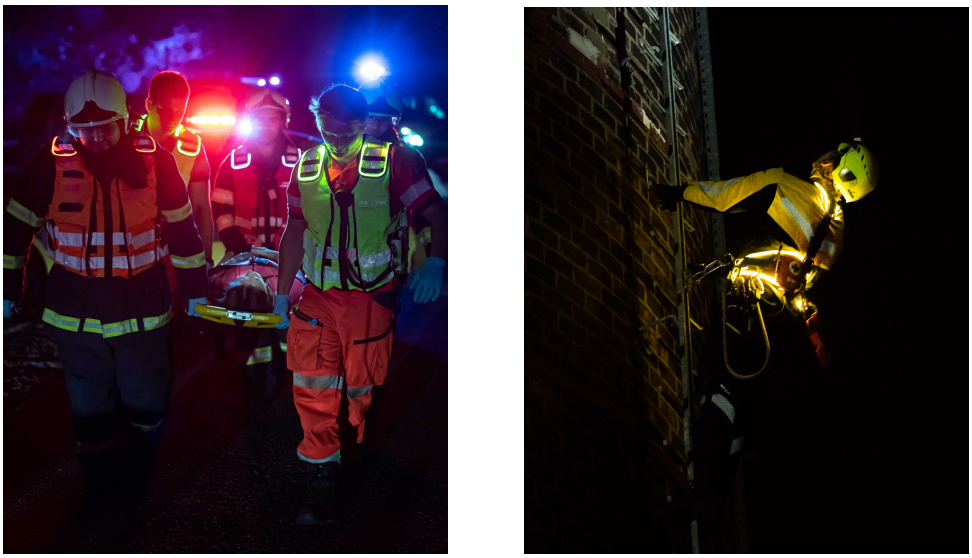


Figure 1.1: SunFibre wearable active lighting system

[1] Taken from SCILIF official website `https://www.scilif.com`

Chapter 2

The Lighting System

This chapter aims at analyzing possible solutions to remote and local control of the SunFibre active lighting system. The introduced principles differ significantly in practicality, reliability, added costs and overall engineering complexity.
Beyond that, it defines and introduces the system devices - **Light Control Unit** and **Floating Button** (names are taken from the client's terminology). Additionally, it explains their purpose as well as user/functional and non-functional requirements.

2.1 Light Control Unit

The light control unit is a battery device that controls the luminous flux emitted by a light guide. The regulation is based on the electrical current flowing through external optical LEDs, which power the lighting system's optic fibres encased in textile clothing (light guide). There are several predefined levels of luminance (brightness) and blinking modes called light modes (see section 2.3). The unit is usually placed in a pocket typically located at the back waist part of a vest or a jacket.

2.2 Floating Button

The floating button is a device placed on a comfortably reachable part of a vest/jacket designed to have good haptic properties. It allows the user to quickly change the light modes on the light control unit. The floating button can be either a wired (then connection cables are needed) or a battery device communicating wirelessly.

2.3 Light Modes

The user's most important interaction with the system is turning on/off the lighting and changing the predefined light modes. At the time of writing, five different light modes according to client requirements were defined:

1. No lighting
2. Strong lighting (130 mA)
3. Low lighting (65 mA)
4. Slow blinking (2 s period)
5. Fast blinking (0.5 s period).

Both blinking modes use strong lighting (130 mA), and the duty cycle 50 %. The state of the art is to develop a satisfying way how to switch these light modes.

2.3.1 Analysis of Methods of Light Modes Changing

Since the spectrum of end-users is large, a particular light mode change method might suit different use cases. When comparing specific methods, the considerations were given especially to these aspects:

- **Practicality**. How easily the light mode can be changed. How much effort is needed. Consider a motorbike driver or a biker use case.
- **Indirect access**. How the light modes can be changed when the unit is not reachable by hands, encased in the clothing or located in a back pocket.
- **Direct/fast switching**. How quickly the light modes can be switched.
- **Reliability**
- **Overall engineering complexity**

Apart from that, client's non-functional requirements must be taken into account too:

- Maintaining low power consumption
- Reducing the additional costs as much as possible
- Limited use of electromagnetic fields in some use cases (civil services).

The table 2.1 below compares multiple light mode change methods according to the criteria defined above.

A downside of many methods (for example, RFID, touch sensors, magnetic relay switch etc.) is that the sensors sensing user activity (to change the light mode) must be wired because they are located on a comfortably reachable part of the clothing. Adding wires and encasing them in clothing brings design concerns and additional costs.

2.3.2 Selected Methods of Light Modes Changing

During the work on the thesis, multiple methods of light mode change were implemented (they are highlighted in table 2.1). The selection of particular methods was based on client preferences.

Principle	Practical	Direct/Fast Switching	Reliable	Engineering Complexity	Power Consum.	Added Costs
Push Button (on the board)	low (no direct access)	high	high	low	very low	no
Magnetic Relay Switch (sewed, wired)	high	low	high	low	very low	medium
Capacitive Touch Sensor (wired)	high	high	low (water)	medium	low	low
Accelerometer (sewed, wired)	low (shake), medium (tap)	low	low	medium	low	medium
RFID (sewed, wired)	high	medium	high	medium	high (excitation)	low
Mobile App (wireless - BLE x WiFi)	low	high	high	high	high	medium
Floating Button (wireless - BLE x WiFi)	high	high	high	high	high	very high
Voice Control (wireless)	high	high	medium	very high	high (processing)	low
Gesture Control (sewed, wired)	medium	high	medium	very high	medium (processing)	medium
Optical Touch Sensor (wired)	high	medium	high	high	medium	low

Table 2.1: Comparison of possible light mode change methods

A direct change to the preferred light mode is possible only through mobile application and gesture control. The other methods simply increment the current light mode, and the user have to iterate over them.

Button press on the light control unit directly brings no overhead in terms of costs and battery usage. Nevertheless, it is the less practical method, and in many use cases (for example, when the unit is sewed in clothing), it is not applicable at all.

Remote change by a smartphone application is based on BLE communication. BLE was prioritized over WiFi, especially due to low power requirements. Initially, the BLE slave (light control unit) needs to be bonded and connected to a central (smartphone).
However, the method's practicality is low, especially when a quick change is desired. One still needs to take out the smartphone from the pocket and open the application. Using smartwatches available on the wrist at any time might address this problem. Nevertheless, there are still use cases (for example, when the users wear gloves or muddy and dusty environments) when this method is not applicable. This approach also increases the overall cost and battery consumption since a BLE-based MCU must be used.

Change by RFID is based on the inductive coupling principle between an RFID tag and a reader. Both of them can be sewed - the coil of the RFID reader into an arbitrary, comfortably reachable part of the clothing and the RFID tag into a sleeve. When the user brings the sleeve with sewed RFID tag close enough to the reader, it triggers the change of a light mode.

This method might be very practical and easy to use, but on the other hand, slow because the light mode can not be changed directly but only incrementally. Furthermore, the biggest drawback is the increased battery consumption required for RFID detection and the necessity to stitch cables onto the clothing between the coil and the RFID reader.

Remote change by tapping on the floating button is based on BLE communication between the central (floating button) and the peripheral (light control unit). Initially, the light control unit must be bonded and connected. No cables are required because the communication is wireless, and the button can be located on an arbitrary, comfortably reachable part of the clothing. Nevertheless, this method brings the most considerable overhead in terms of the costs and overall engineering complexity since two BLE devices must be used.

Change by a gesture, based on data from an accelerometer and a gyroscope, allows the user to change the light modes directly by a predefined gesture. The reliability and practicality of this method are questionable, so it is rather considered an experimental feature.

Last but not least, one should take into account that these methods could be combined (for example, the BLE-based floating button can perform the gesture control etc.), and it always depends on the use case and customer preferences.

2.4 Monitoring & Configuration & FW Upgrade

Apart from controlling the light modes, another important user interaction with the system is monitoring its condition and configuration. The monitoring data and configurations should be available through a mobile application.

Monitoring

Considering that both the light control unit and the floating button are battery devices, the system should provide information about battery level and charging status. Besides that, monitoring the LEDs' voltage, the current flowing through them and temperature is also crucial to quickly identify malfunctions.

Configuration

If the unit uses RFID detection to change the light modes, the user should be able to set the paired tag ID and, in some cases, turn on/off the RFID detection in order to save power.

Firmware Upgrade

The firmware and hardware version should be provided by the devices and available in the mobile application.
In the case of a new firmware version, OTA (Over The Air) firmware upgrade enables the user to flash a new firmware version to the device through a mobile application over BLE. It plays an important role, especially during continuous (agile) development.

2.5 Controlling the BLE Operation

Because both the light control unit and floating button are BLE devices, the user must be able to control the state of the BLE connection.

The required user interactions with the light control unit and floating button are shown below. *LCU* denotes the light control unit, *FB* floating button and *MP* mobile phone (application).

Light Control Unit:

- Connect already paired FB/MP
- Connect and bond new FB/MP
- Disconnect a connected FB/MP
- Clear data of all bonded peers

Floating Button:

- Connect (bond if needed) LCU
- Connect (bond if needed) MP
- Disconnect a connected LCU
- Disconnect a connected MP
- Clear data of all bonded peers

2.6 Non-Functional Requirements

User Interface Design

Coming to one of the non-functional requirements, it is essential to emphasize that consideration must be given to a good and intuitive user interface design. Since the input/output capabilities are on both devices limited, the user interface should make full use of them.

Battery Consumption

Since both system devices are battery devices, it is crucial to achieve a good battery life. Despite the fact that most of the energy is consumed by powering the light guide (the strong light mode uses 130 mA), the battery usage coming from other sources (RFID detection, BLE communication, gesture recognition etc.) shouldn't bring a considerable overhead and must be kept at a minimum. The goal is to achieve months on one battery cycle if the device sleeps and days when active (assuming the lighting is turned off). Hence, appropriate hardware design and code optimization are needed.

Chapter 3

Bluetooth Low Energy & RFID

The goal of this chapter is to provide a theoretical basis for communication technologies used in the project. Because both BLE and RFID are complex technologies, the focus is given to aspects that were directly considered and incorporated during development.
The definitions and parameters introduced here are further referenced in the following chapters with the reasoning behind their selection and setup. For example, the section 5.4 describes the selected BLE configurations.

3.1 BLE

Bluetooth Low Energy (BLE) is a wireless personal area network technology independent of Bluetooth Classic. It is aimed at IoT applications and intended to provide reduced power consumption and cost while maintaining reasonable communication capabilities on short range. BLE defines generic profiles, GAP and GATT, ensuring interoperability between all BLE devices [36].

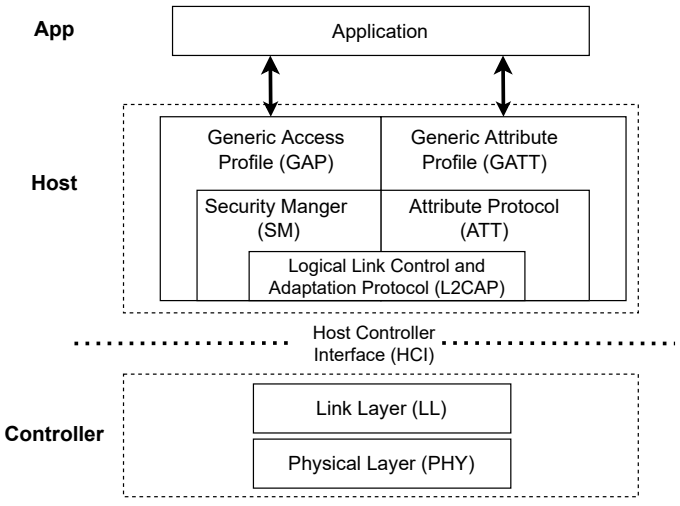


Figure 3.1: The BLE protocol stack

As shown in figure 3.1, the BLE protocol stack is split into multiple layers grouped in three parts: controller, host and application. Some of the controller

layers (PHY and LL), as well as the upper host layers (ATT, SM, GAP and GATT) are introduced in the following sections.

3.1.1 Physical Layer (PHY)

The BLE physical layer is responsible for transmitting and receiving information via radio waves and all operations that are connected with that (encoding, modulation, demodulation, decoding etc.). BLE uses the unlicensed 2.4 GHz ISM (Industrial, Scientific, Medical) band, particularly the frequencies from 2.402 GHz to 2.480 GHz. This band is divided into 40 channels, 37 of these channels are used for connection data and the last three channels for advertising (see figure 3.2).

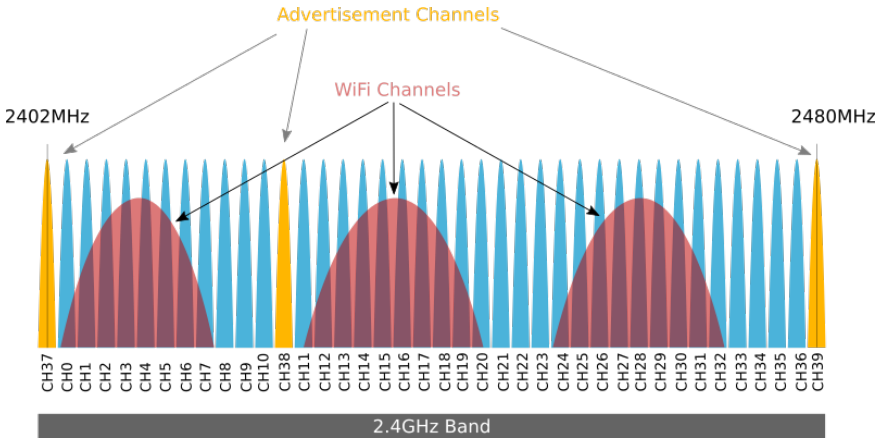


Figure 3.2: BLE advertising and data channels (taken from [2])

BLE adopts the frequency hopping spread spectrum to reduce the effect of potential radio interference (especially from WiFi and classic Bluetooth). Gaussian Frequency Shift Keying (GFSK) modulation is used in BLE.

The data bitstream is at first encoded by non-return to zero (NRZ) sequence, where the logical one is represented by a positive voltage and the logical zero is represented by a negative voltage. The sequence is further filtered by a Gaussian filter which helps to reduce the sideband power, thus decreasing the potential interference with the neighbouring channels. The sequence is then modulated by an FSK modulator and finally transmitted on the desired channel.

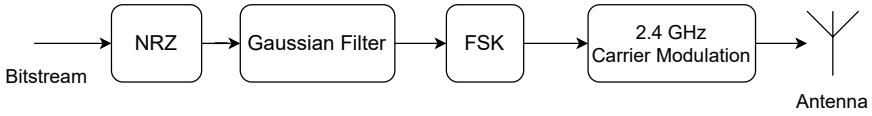


Figure 3.3: GFSK modulation

The modulation (data) rate of Legacy BLE (version 4) is fixed at 1 Mbit/s, whereas the BLE Version 5 achieves 2 Mbit/s. Therefore, it is the upper physical throughput limit of the technology [36, p. 17].

3.1.2 Link Layer (LL)

The BLE link layer is the only hard real-time constrained layer of BLE since it must comply with all the timing requirements defined in the specification. It is in charge of packet preparation - assembly, CRC generation, data whitening, encryption, etc. It also performs error detection and correction.
BLE has only one packet format - PDU (Protocol Data Unit) and two types, either the Advertising Channel PDU or the Data Channel PDU. Before encoding, the data length ranges between 2 and 257 bytes.

The BLE link layer defines the following roles [36, p. 18]:

- Advertiser - device sending advertising packets
- Scanner - device scanning for advertising packets
- Master - device initiating a connection
- Slave - device accepting a connection

Advertising and Scanning

BLE devices are detected by broadcasting advertising packets. The advertising device sends a packet on at least one of the three advertising channels. The advertising packets can be either directed, containing the target scanner's Bluetooth address in its payload, or undirected, not targeted to any particular scanner.

Advertising interval denotes how often the advertising packets are broadcasted. It ranges from 20 ms to 10.24 s. The shorter the advertising interval, the more power the device consumes since the radio has to wake up more frequently. On the contrary, it leads to a higher probability of packets being scanned and thus faster discoveries.

Scan interval and *scan window* define how often and for how long a scanner listens for potential advertising packets. Because the advertiser uses three different channels and there is no synchronization with the scanner, the advertising packet is successfully scanned only if the used channels overlap.

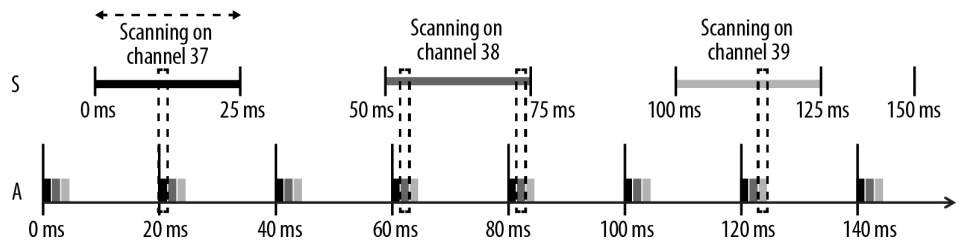


Figure 3.4: Advertising and scanning (taken from [36])

The figure 3.4 shows advertising with 20 ms advertising interval, 50 ms scanning interval and 25 ms scan window.

Active Scanning

The advertiser requests a connection by periodically transmitting a connectable `Avertising Indication` packet. After receiving the advertising indication, the scanner issues `Scan Request` packet, and the advertiser replies with `Scan Response` packet.

Passive Scanning

The advertiser transmits non-connectable `Avertising Indication` packets, usually periodically, and it is never aware whether it was received by any scanner [36, p. 20].

Advertising PDU

The advertising indication, scan request and scan response packets are all types of advertising PDU. According to Bluetooth Core Specification version 5.2, they consist 2-byte header determining the packet type and length and up to 255 bytes of user data [5, p. 2871]. The user data is made up of advertising data (AD) elements defined by Bluetooth SIG and structured as length - type - value.

Connections

The connection procedure starts when a master scans for an advertiser that accepts connection requests (transmits connectable advertising packets) and sends `Connection Request` packet. The established connection between master and slave is kept by regular packet exchanges at predefined times - connection events. Despite the fact that there is no application data to send or receive, it is still required to exchange at least the empty link layer packet with no payload (Empty PDU) to maintain the connection.

The advertising packets can be filtered based on Bluetooth address or by advertising data - for example, a BLE service UUID. Connection requests can be filtered by a white list, an array of Bluetooth device addresses that are allowed to connect [5, p. 2933].

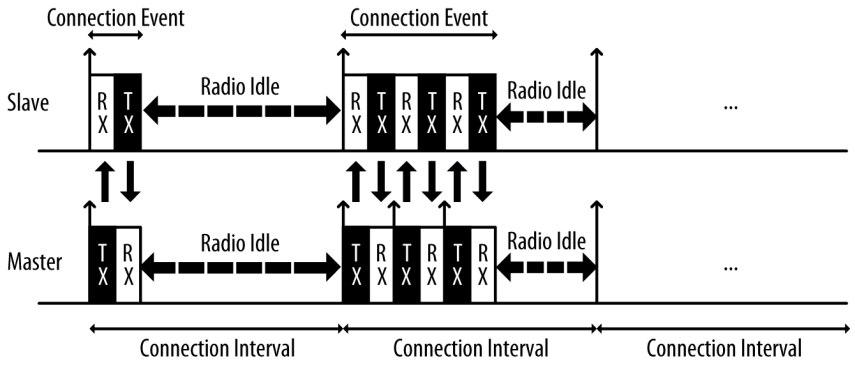


Figure 3.5: Connection events (taken from [36])

Connection Parameters

The connection parameters for a BLE connection are a set of properties of the link that determine how and how often the central and peripheral communicate. These parameters are a key factor in balancing throughput and power consumption. The parameters are exchanged when the connection is initially established, and they are always determined by the central. However, the peripheral can send a `Connection Parameter Update Request` for changing them.

Connection Interval is the time between the beginning of two consecutive connection events. It ranges from 7.5 ms to 4 seconds.

Slave Latency is the number of events that a slave can skip if he doesn't have data to send without risking disconnection.

Connection Supervision Timeout is the maximum time between two received data packets before a connection is considered as lost [36, pp. 22–23].

Throughput Parameters

The throughput parameters determine how much data can be exchanged between devices during a connection event.

Event Length gives the maximum length of connection intervals.

Data Length is the size of the usable payload of data PDU.

MTU Size is the maximum transmission unit of ATT protocol - payload size of ATT packets.

3.1.3 GAP

GAP is the topmost control layer of BLE. It dictates how devices interact with each other - defines roles, modes of operation, security and procedures (for example, device discovery, connection establishment, connection parameter update, etc.)

GAP specifies four roles that the devices can adopt. The roles correspond to the link layer roles defined above (3.1.2):

- Broadcater - advertiser, transmit-only applications
- Observer - scanner, receive-only applications
- Central - master, capable of multiple connections establishment
- Peripheral - slave, allowing connection establishment with masters

The BLE protocol is asymmetric, meaning that the computational requirements for masters are larger than for slaves. This problem is mitigated because the central role is usually performed by more powerful devices with larger energy sources (smartphones, tablets or computers), whereas the peripherals are battery devices with less processing power.

Security Manager (SM)

SM is both a protocol and a set of security algorithms to generate and exchange security keys for link encryption. GAP further adds security procedures and modes allowing the implementation of security measures [36, pp. 45–47].

Pairing

Pairing involves a generation of temporal Short-Term security Key (STK) for link encryption. The temporal key is not preserved and can not be reused in future.

Bonding

Bonding comprises pairing - encrypting the link using an STK and then distributing a Long-Term Key (LTK) for faster security establishment in the future. The LTK is a 128-bit key shared by both devices and destinated to be stored in nonvolatile memory.

Pairing Methods:

- Just Works - exchange of STKs in plain text
- Passkey Display - one of the peers displays a randomly generated passkey
- Out of Band - transfer of additional data by other technologies (NFC)
- Numeric Comparison

The just works method is the weakest pairing method, providing no security against man-in-the-middle attacks, compared to the other methods. It is used when the devices have no input/output capabilities (display and keyboard).

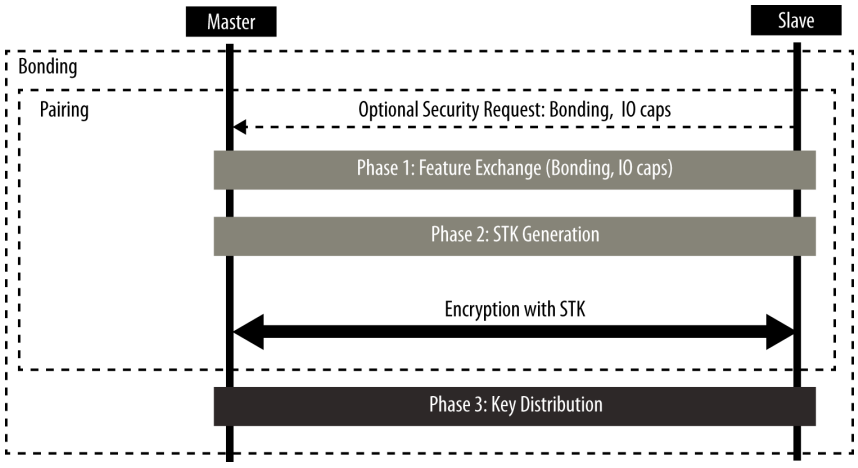


Figure 3.6: Pairing and bonding sequences

GAP Security Modes

For a BLE connection, GAP defines two security modes with several security levels. Security mode 1 enforces the security by encryption, whereas mode 2 by data signing.

Security mode 1:

- Level 1 - no security (no encryption)
- Level 2 - unauthenticated encryption (Just Works pairing method)
- Level 3 - authenticated encryption (stronger pairing method)
- Level 4 - authenticated encryption with LESC (strongest security)

Security mode 2:

- Level 1 - unauthenticated data signing
- Level 2 - authenticated data signing

Security mode 1, level 1 is the default mode for every connection, but it can always be upgraded depending on the GATT attribute permissions and application requirements.

3.1.4 GATT

The GATT (Generic Attribute Profile) specifies procedures and formats for data exchange over a BLE connection. GATT defines these roles:

- Client - device that sends requests to a server and receives responses and server initiated updates
- Server - device that makes the user data organized in attributes available to the client

GATT roles are independent of GAP roles, meaning that both a GAP central and a GAP peripheral can act as both GATT client or server or both, even simultaneously

Attribute Protocol (ATT)

GATT is built on a more generic Attribute Protocol. ATT defines how data are represented in a server's database and the actions that a client and server perform to access them - R/W requests and responses, commands, notifications, indications and confirmations. The user data are organized in the form of attributes - the smallest data entities, each represented by an attribute handle, a type defined by a UUID, a set of permissions and a value.

Attribute Handle

The attribute handle is a unique 16-bit identifier of each attribute on a GATT server. A service and characteristics discovery is needed in order to obtain the handles to reference particular attributes.

Attribute Type

The attribute type is represented by a UUID, a 128-bit number that is guaranteed to be globally unique. BLE adds two additional UUID formats - 16-bit and 32-bit that can be only used for UUIDS reserved by Bluetooth SIG.

Attribute Permissions

The allowed ATT operations are specified by permissions. The list of possible permissions is shown below.

Access Permissions:

- None
- Readable
- Writable
- Readable/writable

Encryption:

- No encryption (SM 1, level 1)
- Unauthenticated encryption (SM 1, level 2)
- Authenticated encryption (SM 1, level 3)

Attribute Value

The actual data content of an attribute is referred to as attribute value. The attribute value can have an arbitrary length of up to 512 bytes.

GATT Hierarchy

GATT adds a layer of hierarchy and data abstraction on the top of ATT by grouping the attributes as characteristics of services within profiles. The whole set of attributes of a GATT server can be visualized as a table.

GATT Services

GATT service groups logically related attributes organized in characteristics into one common section. A service can have one or more characteristics, and each service distinguishes itself from another by UUID. A primary service declaration attribute type is always a 16-bit SIG-assigned UUID (**0x2800**).

GATT Characteristics

A GATT characteristic is a container for a piece of user data. It contains the actual data - *Characteristic Value Declaration* attribute, along with metadata - *Characteristic Declaration* attribute. Additionally, the characteristic can include *Descriptor Declaration* attributes, which further extend the metadata containing, for example user description, client configuration etc.

The attribute type of a characteristic declaration is a standard 16-bit UUID (**0x2803**). Its attribute value contains a handle, a UUID, and a set of properties. These three elements describe the following characteristic value declaration. The handle points to the characteristic value declaration's place in the attribute table. The UUID describes what type of information is expected, and the properties describe how the characteristic value can be interacted with.

The attribute type of characteristic value declaration is the same arbitrary UUID (16-bit for SIG-recognized or 128 bit for custom characteristics) as specified in the attribute value of a characteristic declaration. The actual value of the characteristic (for example measured temperature), is located in the attribute value.

3.2 RFID

Radio-frequency identification (RFID) is a wireless communication technology that enables uniquely identify tagged objects or people. An RFID system is composed of at least two components - a tag and a reader. Sometimes, the system is controlled by a host computer [7].

3.2.1 RFID Tags

The basic function of an RFID tag (sometimes called a transponder) is to store data and transmit them to the reader. An RFID tag contains a tiny semiconductor chip and miniaturized antennas in some form of packaging (for example, a key fob). The tag and the reader communicate over radio waves. When a tagged object enters the read zone of a reader, the tag starts transmitting its stored data. Tags can hold many kinds of information about the objects they are attached to - serial numbers, timestamps, configuration etc [7].

Some RFID tags are referred to as **active tags**, including an energy source such as small batteries. When the tag needs to transmit data to the reader, it derives the power for the transmission from this energy source. Hence, the active tags can communicate with less powerful readers and can transmit information over much longer distances. Furthermore, these types of tags typically have larger memories. However, they are also much larger and complex, and thus expensive [7].
The **passive tags** don't have any on-board power source. Instead, they derive the power for the data transmission from the energy of electromagnetic field created by a reader. As a result, the passive tags are typically smaller and less expensive than active tags. On the other side, the read range of passive tags is much shorter, they require more powerful readers and have less memory capacity.

Some RFID tags are **read-only** - factory preprogrammed containing a limited amount of data (typically only a serial number). Their biggest advantage is a low price.
Conversely, their **read/write** counterparts contain more memory providing the user with more flexibility and the opportunity to be reused. They have an addressable memory that can be easily rewritten.

3.2.2 RFID Readers

The most important functions of an RFID reader (sometimes called an interrogator) are to read data from an RFID tag, power a tag (in case of passive tags) and write data to a tag (in case of smart tags). An RFID reader contains an antenna, RF circuit and controller electronics. In addition to the basic functions described above, the more complex RFID reader can also perform the anti-collision measures allowing communication with multiple tags simultaneously, tag authentication and data encryption.

3.2.3 RFID Frequency Bands

The RFID uses multiple frequency bands. The frequency used affects the properties of an RFID system (such as read range, data rate, etc.).

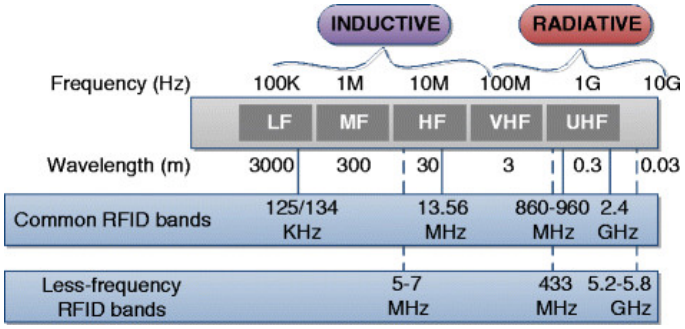


Figure 3.7: Frequency bands of an RFID system (taken from [27])

Low Frequency RFID Bands:

- Low frequency (LF): 125-134 kHz
- High frequency (HF): 13.56 MHz

High Frequency RFID Bands:

- Ultra-high frequency (UHF): 860-960 MHz
- Microwave: 2.5 GHz and above

Low Frequency Bands

Because of the long wavelengths of low-frequency radio signals, the antennas of LF and HF RFID tags must be much larger in order to achieve the same signal gain as UHF tags. However, this goes against the goal of making RFID tags small and cheap. Hence, a compromise between the tag size and read range must be made. As a result, the communication range is usually no more than tens of centimeters, mainly due to low antenna gains. The passive tags are commonly operated in low-frequency bands.

High Frequency Bands

The UHF tags benefit from the possibility of being used with much smaller antennas. On the other hand, not only electromagnetic interference but also close obstacles (metal, water etc.) significantly reduce their communication range.
Typically, the reader's range is around a few meters in the case of passive tags and up to hundred meters when active tags are used. The production costs of UHF tags are usually too high to be massively applied. The active tags are usually operated in high-frequency bands [27].

3.2.4 RFID Data Transmission

In RFID, the data transmissions are realized by coupling - an energy transfer over the electromagnetic field. The passive tags use coupling to obtain power and transfer data. The RFID systems use two types of coupling depending

on the frequency and the distance between the tag and the reader's antenna - inductive or backscatter.

Inductive Coupling

Inductive coupling uses near-field effects - a transfer of energy from one coil to another through a shared magnetic field (see figure 3.8). Generally, the LF and HF RFID systems using solenoid coils employ inductive coupling.

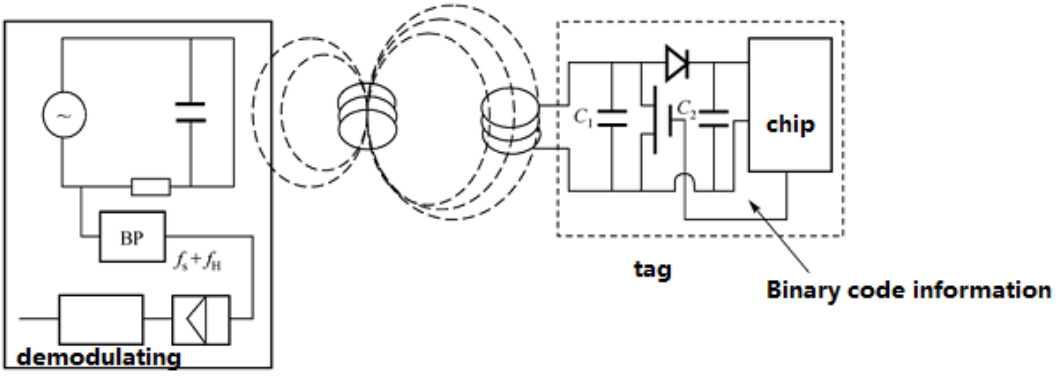


Figure 3.8: Inductively coupled RFID system (taken from [30])

Backscatter Coupling

Backscatter (radiative) coupling uses far-field effects. A tag responds to a reader signal or field by modulating and re-transmitting the response signal at the same carrier frequency. The tag antennas are designed to resonate well with the carrier signal emitted by a reader. UHF and microwave RFID systems use backscatter coupling [7].

The periodic signal transmitted by an RFID reader does not carry any information. To transmit tag's data, the carrier signal must be changed, data modulated on the signal. The RFID signals use analog carrier and are digitally modulated. These possible modulations are used:

- Amplitude Shift Keying - ASK
- On-Off Keying - OOK (type of ASK)
- Phase Shift Keying - PSK
- Frequency Shift Keying - FSK
- Pulse Interval Encoding - PIE.

Chapter 4

HW Specification

This chapter serves as a description of hardware components and functional blocks used in the system. Attention will be given to the technical details of specific components and, for some, the reasons behind their selection.

4.1 HW of Light Control Unit

The light control unit board is based on the nRF52833 BLE SOC (System on Chip) from Nordic Semiconductor (4.1.1). The board also contains a stabilizer circuit, charger circuit (4.1.5), battery measurement circuit (4.1.3), USB-C connector, three small RGB LEDs for signalization and one push button.

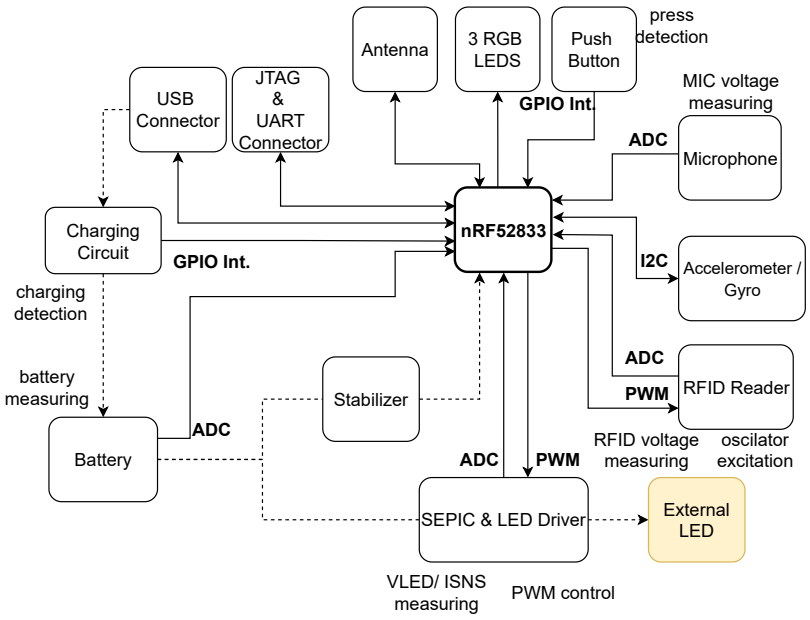


Figure 4.1: Block diagram of the light control unit

To regulate the luminance of the light guide, the electrical current flowing through the diodes must be controlled by a LED driver (4.1.4). The external

LEDs illuminating the optic fibres can be connected to the board through a 2.5 mm jack. Furthermore, the board also contains an accelerometer/gyroscope LSM6DSMTR (4.1.6) and a microphone. The 1600mAh 3.7V Lithium-ion rechargeable battery is soldered directly on the PCB and needs to be bridged by a jumper. A Molex 2.4/5 GHz linear, omnidirectional antenna with a peak gain of 2.4 dBi can be connected to increase the reach.

4.1.1 Microcontroller (MCU)

The nRF52833 is general-purpose multi-protocol SOC (System On Chip) device supporting not only BLE but also Zigbee, Bluetooth Mesh and NFC.

Specification of nRF52833 with selected peripherals:

- 32-bit RISC ARM Cotex M4 with FPU, 64 MHz
- 512 kB flash and 128 kB RAM memory
- HFCLK from 64 MHz on-chip oscillator / 32 MHz external oscillator, LFCLK from 32 kHz RC oscillator
- 42 GPIO pins
- 12-bit ADC 200 kSa/s, 8 channels
- 4x PWM, 4 channels
- 2x UART, 4x SPI, 2x I^2C
- 5x 32-bit timer with counter mode, 3x RTC counter
- temperature sensor

Power Management:

- Power consumption in SYS-OFF mode: 0.6 µA (at 3V)
- Power consumption in SYS-ON mode (IDLE): 1.5 µA (at 3V)
- Power consumption when computing: 5 mA (at 3V)
- 4.9 mA peak current in TX (0 dBm)
- 4.6 mA peak current in RX [24]

Alternatives

Even though Nordic is the leading company in the field of BLE SoCs, there is some competition. Eventually, the nRF52 series was chosen because of positive previous experience with the chip predecessor nRF51.

TI CC2650 is a BLE chip from Texas Instruments based on ARM Cortex-M3 running at 48 MHz, cost-effective and low power. Unlike the nRF, it has a separate chip for BLE stack.

ESP32 is a WiFi/BLE chip from Espressif running at a fast, 240 MHz dual-core Tensilica LX6. It is inexpensive but not designed for low power applications (peak currents around 100 mA).

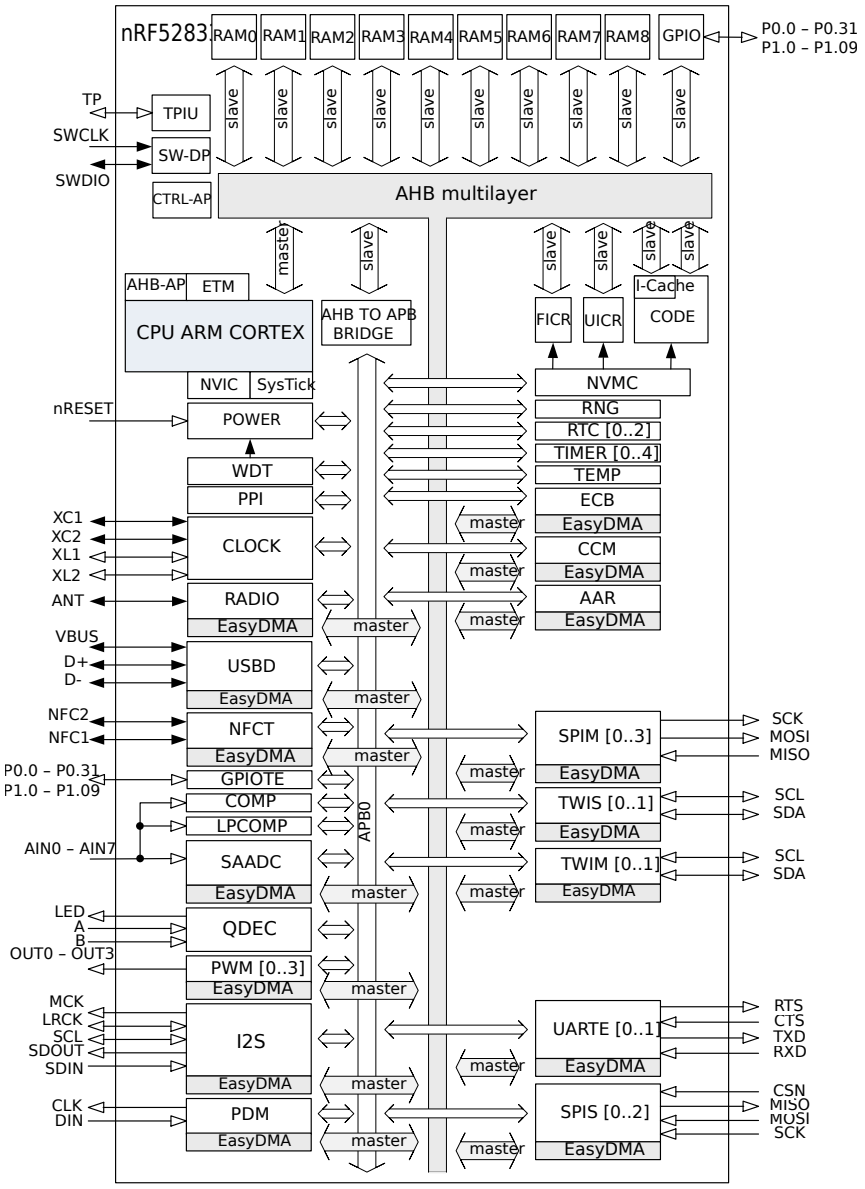


Figure 4.2: nRF52833 block diagram (taken from [23])

BTLC1000 is a BLE chip from Atmel based on 26 MHz ARM Cortex-M0, ultra-low-power (TX, RX peak currents are not higher than 4 mA). Apart from the speed, the disadvantage is that it doesn't incorporate flash memory.

PPI

PPI (Programmable Peripheral Interconnect) is an advanced chip's interface that allows peripherals to connect and communicate with each other using a system of tasks and events without the MCU involvement. It becomes beneficial, especially when precise synchronization between events and tasks

is needed (for example, precise ADC sampling).

EasyDMA

EasyDMA is easy to use Direct Memory Access that provides some peripherals with direct access to the RAM without the need for the MCU. Multiple DMA instances (channels) for a peripheral are allowed, so a peripheral can write to the RAM (for example, measured data) and read from it simultaneously.

ADC

The successive approximation analog-to-digital converter integrated in nRF52 supports up to eight external input channels (8 single-ended inputs or 4 differential inputs) in 8/10/12 bit resolution. It uses EasyDMA to store the sampled values directly to the RAM.

Each ADC channel can use an individual voltage reference - either the 0.6 V internal reference or VDD/4. Using the VDD based reference is useful if the measured signal fluctuates depending on VDD. The channel gain setting ranges from 1/6 (6x amplified) to 4.

The maximum sampling frequency is 200 kHz (inverse value of summation of conversion time - $t_{conv} = 2\,\mu s$ and minimum acquisition time - $t_{acq} = 3\,\mu s$. The acquisition (sampling) time determines how long the channel input is connected to the external source.

4.1.2 Radio

The 2.4 GHz transceiver is integrated on the nRF52833 chip. It supports multiple radio standards such as 1/2 Mbps BLE modes, long range 250/500 kbps BLE modes or IEEE 802.15.4 based 250 kbps mode (LR-WPAN). Besides that, it also implements the Bluetooth direction finding.

The radio uses DMA for reading and writing data packets from and to the RAM without CPU involvement. Apart from that, it involves an automated packet assembler, packet disassembler, automated CRC generator and CRC checker.

Specification of radio:

- 1 and 2 Mbps BLE modes
- 250 and 500 kbps long range BLE modes
- IEEE 802.15.4 250 kbps mode (LR-WPAN)
- -96 dBm sensitivity in 1 Mbps BLE mode
- -103 dBm sensitivity in 125 kbps BLE mode (long range)
- -20 to +8 dBm TX power, configurable in 4 dB steps

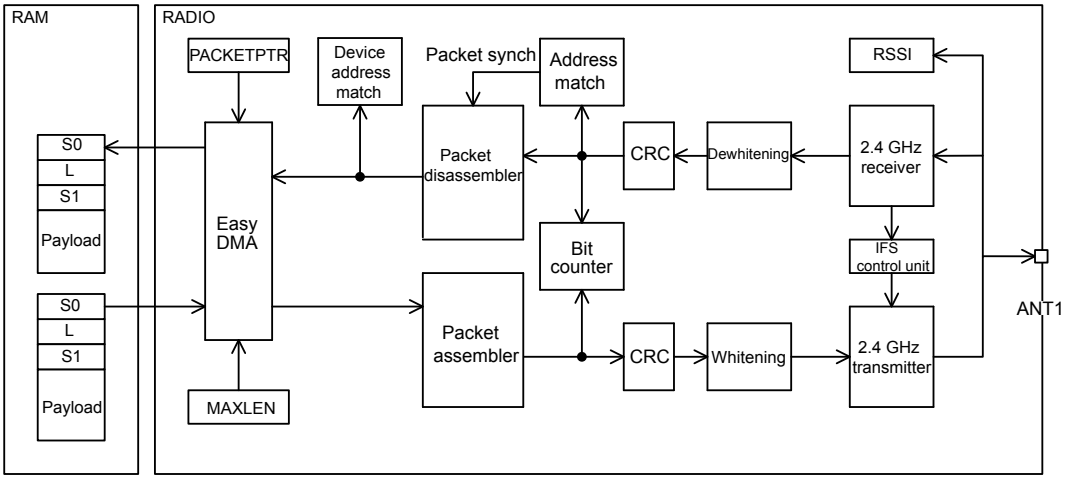


Figure 4.3: nRF52833 radio block diagram (taken from [25])

4.1.3 Battery Measurement Circuit

The lithium battery has typically a voltage range of 2.7 - 4.2 V [12].

This subsection provides an analysis of the circuit for battery measurement. It consists of a voltage bridge and capacitor (see figure 4.4). The component selection plays an important role in ADC setup (described in section 5.9.1).

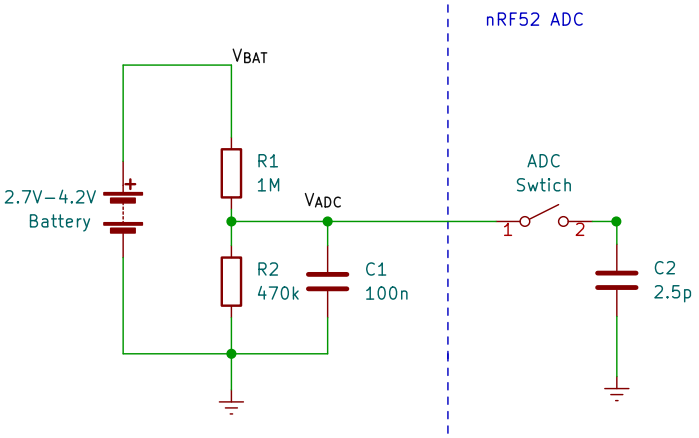


Figure 4.4: Battery measurement circuit

Leakage Current

Because the battery is connected permanently, the aim is to reduce the leakage current to the minimum, so the total resistance of the voltage divider should be as high as possible. According to the equation 4.1, the maximum value of the leakage current (when the battery is fully charged) is $I_{leak_{MAX}} = 2.857\,\mu A$. Considering the capacity of lithium batteries powering the units - 1600 mAh, the effect of the leakage current becomes negligible.

$$I_{leak} = \frac{V_{BAT}}{R_1 + R_2} \tag{4.1}$$

Input Range

The ADC input range can be calculated according to equation 4.2 assuming the typical range of lithium batteries. The maximum value helps to determine the ADC gain (see table 5.8).

$$V_{ADC} = V_{BAT} \frac{R_2}{R_1 + R_2} \tag{4.2}$$

$$V_{ADC_{MIN}} = 863.3\,\text{mV}$$
$$V_{ADC_{MAX}} = 1342.9\,\text{mV}$$

Acquisition Time & Voltage Drop

Using the capacitor in the circuit has several advantages, but most importantly, it allows setting down the acquisition time. Without the external capacitor, the internal ADC capacitor C_2 would have to be charged with a large RC time constant (due to the large value of R_2). Hence, a long acquisition time would be needed.
The external capacitor C_1 is fully charged when the ADC input is disconnected. At the start of sampling, the electrical charge is distributed between external and internal capacitors proportionally to their capacities. Therefore there is a voltage drop on the external capacitor (see equation 4.4). The capacitor value C_2 should be high enough to provide the desired voltage drop of less than 1 LSB.

The maximum voltage drop can be calculated assuming the battery is fully charged and the internal ADC capacitor is fully discharged:

$$Q = V_{ADC_{MAX}}\, C_1 = V\, C_1 + V\, C_2 \tag{4.3}$$

$$\Delta V = V_{ADC_{MAX}} - V = V_{ADC_{MAX}} \frac{C_2}{C_1 + C_2}. \tag{4.4}$$

The variable V is the voltage on capacitors after the electrical charge is distributed.

The value of the LSB can be calculated by knowing the maximum input voltage - V_{RES} and the number of bits as follows:

$$1\,LSB = \frac{V_{RES}}{2^n}. \tag{4.5}$$

The resulting maximum voltage drop for selected battery channel gain - 1/3 and the resolution - 10 bit is $\Delta V = 3.4\,\mu\text{V}$ which is less than $1\,LSB = 1.8\,\text{mV}$

4.1.4 Constant Current LED Driver with SEPIC

This subsection shortly introduces the electronics design of the external LED current regulation and thus light guide luminance. The regulation circuit is based on a SEPIC circuit and a DIO5661 chip controlled by PWM from nRF52.

SEPIC (Single-Ended Primary Inductor Convertor)

A SEPIC is a type of DC-DC voltage converter that is able to both step-up (boost) and step-down (buck) the input voltage. The SEPIC is based on a boost converter (L1, S1, D1 and C2), but the circuit contains two inductors. One is connected to the input and the second one to the ground, both connected by a coupling capacitor (see the schematics 4.5).
The SEPIC circuit has advantages over other converters, namely non-inverted output and powered device protection (output voltage is zeroed in case of a switch short circuit). The output of the SEPIC is controlled by the duty cycle of the switch (DIO5661 in our case).

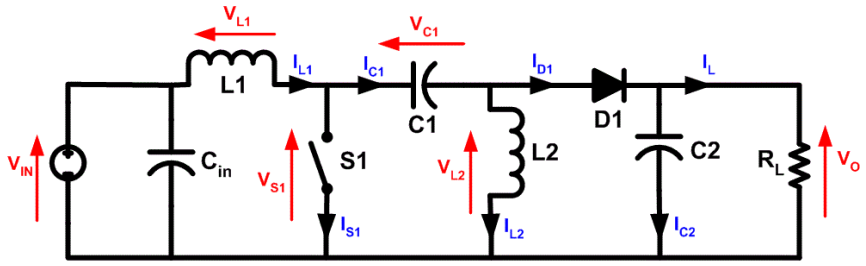


Figure 4.5: Typical SEPIC topology (taken from [32])

The ability to either buck or boost voltage plays an important role because of two aspects. First, the required output voltage - *VLED* varies (one or two LEDs with different forward voltage drops can be connected) and second, the lithium battery voltage changes significantly as the battery discharges. Regardless of the input voltage varying above or below the required output LED voltage, the SEPIC maintains the output voltage and LED current - *ISNS* constant.

LED Driver DIO5661

DIO5661 is PWM to constant current converter. It works on a high 1.1 MHz internal oscillator switching frequency. As illustrated in the functional block diagram 4.6, a gate drive block periodically opens a power MOSFET synchronously with the internal oscillator and closes it according to the Current Sense.

The voltage on an external feedback resistor is compared with an internal voltage reference. This output difference is then amplified in an error amplifier and compared with Current Sense [3]. To sum up, the voltage on the external feedback resistor (R1 in diagram 4.7) determines for how long the transistor

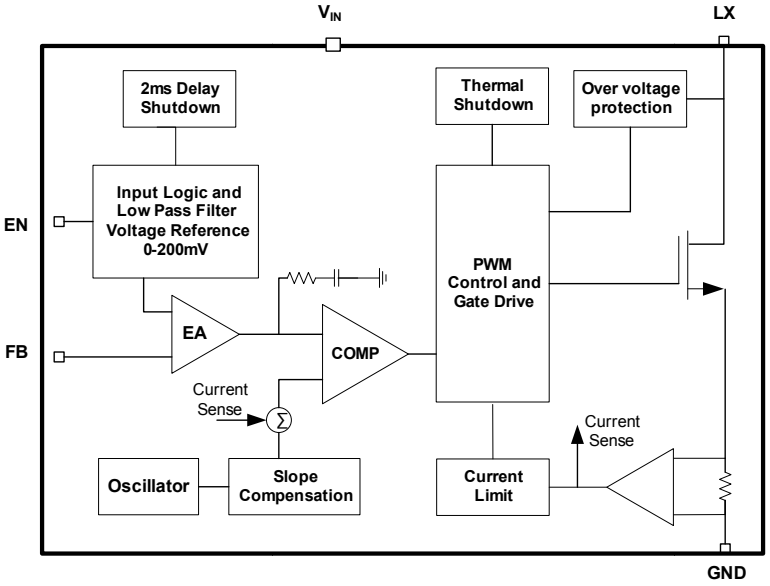


Figure 4.6: DIO5661 functional block diagram (taken from [3])

is opened and the closed-loop stabilizes the voltage on the feedback resistor.

The reference can be set externally by the PWM duty cycle. The PWM signal comes from the nRF52 GPIO pin and is filtered on a low pass filter and its mean value is used for comparison. The allowed PWM frequency range is between 200 Hz to 200 kHz.

The allowed feedback voltage ranges from 0 to 200 mV and its value is proportional to the duty cycle of PWM signal. While using $R_{FB} = 1.5\,\Omega$ ($R1 - SEP$ in figure 4.7), it gives the maximum LED current approximately $I_{FB} = 133.3\,\text{mA}$ and thus sensitivity $\Delta I_{FB} = 1.333\,\text{mA}$. I_{FB} is referred as $ISNS$ in other sections.

The circuit (figure 4.7) also incorporates a 10V Zener diode. When the external LEDs are disconnected and the output voltage reaches 10 volts, the current starts flowing through the diode, which increases the voltage on the feedback pin and thus decreases the output of the converter.

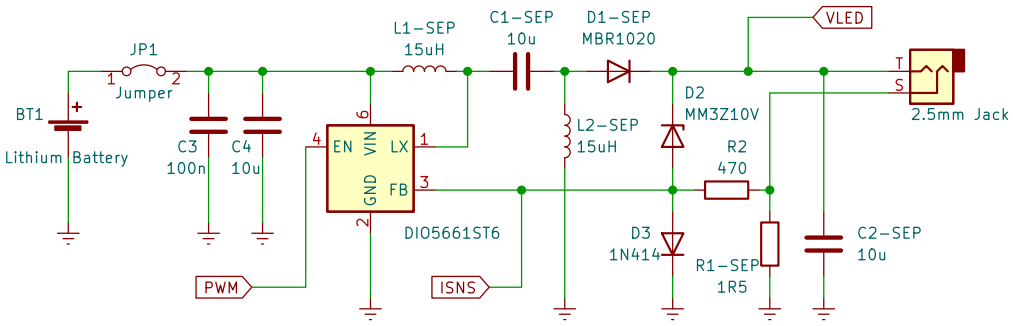


Figure 4.7: LED driver circuit

4.1.5 Battery Charger Circuit

The battery can be charged through a USB-C connector and it is controlled by an additional chip from Microchip MCP73832-2ACI/OT.

It is a linear charge management controller with preconditioning employing a constant-current charging if the battery voltage is below the regulated voltage and a constant-voltage charging if the battery voltage is above the regulated voltage [11]. The constant voltage regulation is factory set (in our case to 4.2 V, corresponding to the maximum voltage of lithium batteries). The constant current value can be selected by an external programming resistor, and it was set to approximately 450 mA.
The logical level of charge status output (STAT) pin determines whether the battery is being charged or not and is connected to the nRF GPIO pin.

4.1.6 Gyroscope/Accelerometer

The LSM6DSMTR is a system-in-package from ST Microelectronics, containing a 3-axis digital accelerometer that is able to measure both DC and AC accelerations, and a 3-axis digital gyroscope, both based on MEMS technology using a 16-bit internal ADC converter. The sensor should be suitable for motion sensing, gestures and orientation detection while maintaining low power consumption [9].

Specification of LSM6DSMTR:.

- Acc. measurement range: selectable from ± 2 to $\pm 16\,g$
- Acc. Sensitivity: from 0.061 mg/LSB to 0.488 mg/LSB (based on range)
- Acc. data rate: selectable from 1.6 Hz to 6.66 kHz
- Gyro measurement range: selectable from ± 125 dps to ± 2000 dps
- Gyro sensitivity: from 4.38 mdps/LSB to 70 mdps/LSB (based on range)
- Gyro data rate: selectable from 12.5 Hz to 6.66 kHz
- Digital filter bandwidth selection
- I^2C or SPI communication
- Acc. power consumption: 4.5 - 160 µA (based on ODR)
- Acc. + Gyro power consumption: 0.4 - 0.65 mA (based on ODR)
- Shutdown current: 3 µA

4.1.7 The Overall Design

The first version of the hardware of the light control unit was designed by Petr Porazil from PiKRON company. The whole scheme is shown in appendix - A.2). The unit's case and sticker come from SCILIF company directly (see figure 4.14).

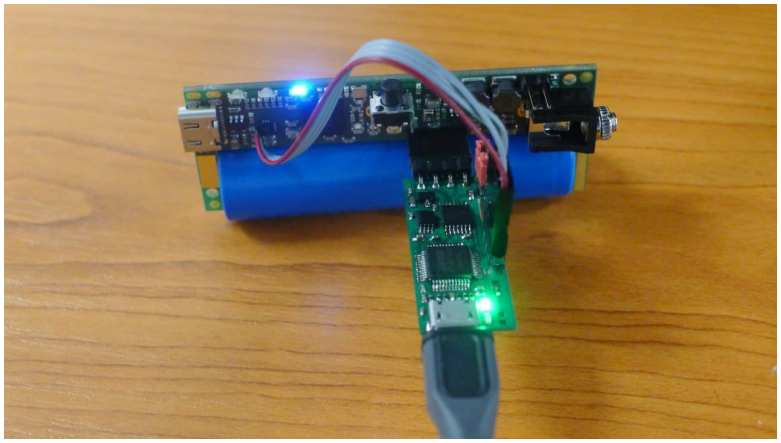


Figure 4.8: Light control unit board from Petr Porazil with PiKRON SWD programmer and USB-UART converter

4.2 HW of Floating Button

For the first prototype of the floating button, the development board BBC micro:bit.v2 was selected. The fact that the board is based on the nRF52833 as well helped significantly during programming.

The board comprises LED matrix of 25 LEDs, two push buttons, a micro-USB connector, two signalization LEDs, a microphone, speaker, touch sensor and an accelerometer/magnetometer LSM303AGRTR.

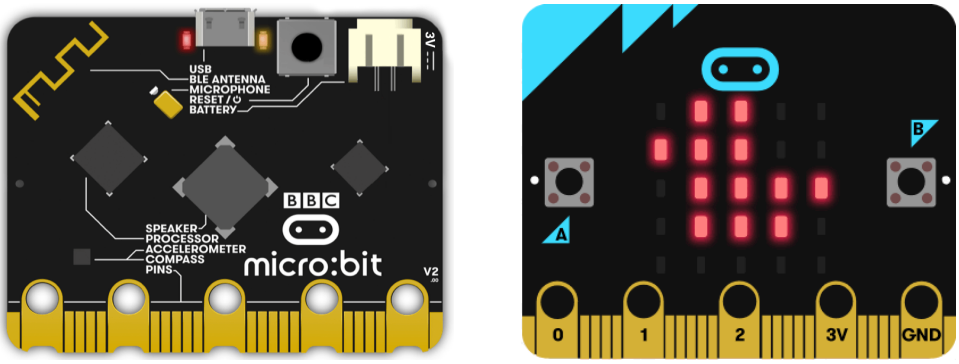


Figure 4.9: BBC micro:bit.v2 (taken from [13])

4.2.1 Accelerometer/Magnetometer

The LSM303AGR is an ultra-low-power system-in-package from ST Microelectronics, containing a 3-axis digital accelerometer and a 3-axis digital magnetometer, both based on MEMS technology. The sensor is not directly intended for gesture control but rather orientation detection, compasses etc. However, at least the accelerometer could be used to detect basic motions [8].

Specification of LSM303AGR:.

- Acc. measurement range: selectable from ± 2 to $\pm 16\,g$
- Acc. Sensitivity: from 0.98 mg/LSB to 187.58 mg/LSB (based on range and power mode)
- Acc. data rate: selectable from 1 Hz to 5.37 kHz
- Digital filter bandwidth selection
- I^2C or SPI communication
- Acc. power consumption: 3.7 - 50 µA (based on ODR)
- Shutdown current: 2 µA

4.3 HW of RFID Reader and Tag

Based on the requirements - low-cost, low data rate, an active RFID reader and a passive RFID tag communicating over low-frequency RFID at 125 kHz were selected. The RFID reader can be integrated as a part of the light control unit board or connected and powered externally.

4.3.1 RDM6300 RFID Reader

For the first prototype of the RFID reader, the development module RDM6300 was chosen. RDM6300 is an LF RFID reader designed to read data from 125 kHz, EM4100 compatible tags using an RF receiver circuit and built-in MCU. An external antenna needs to be connected to the board directly. The RDM6300 uses demodulation and decoding algorithms and only transmits the received bits over a serial communication (UART) at a 9600 baud rate. The whole module was connected to the light control unit through jumper wires.

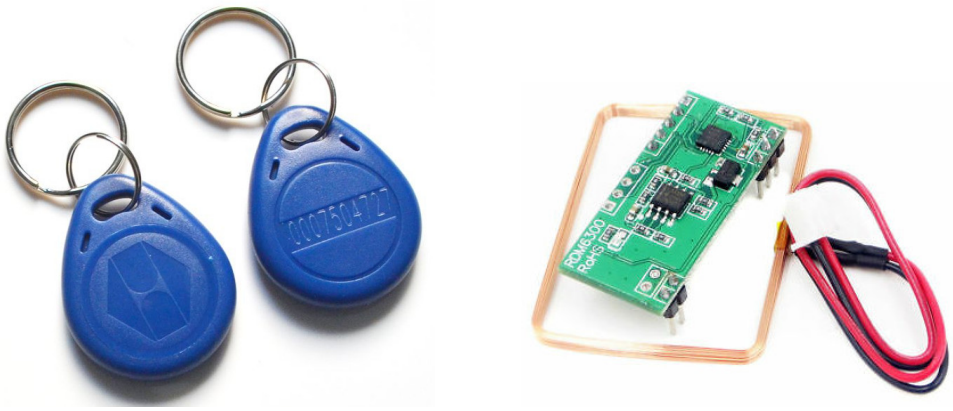


Figure 4.11: EM4100 RFID tag and RDM6300 RFID reader (taken from [35], [1])

Specification of RDM6300:

- 125 kHz operating frequency
- 20-50 mm receive distance
- 5V working voltage
- <50 mA standby current consumption

4.3.2 Custom Low Power RFID Reader

The main drawback of the RDM6300 RFID reader lies in its high standby current consumption (up to 50 mA). As a result, a custom low power RFID reader was designed to mitigate this problem. In the following sections, it is referred to as LC Reader.
The implementation is based on synchronization between excitation of an LC resonance circuit at 125 kHz and synchronous ADC sampling of the carrier to demodulate and decode the RFID signal.

LC Oscillator Circuit

This subsection analyzes the circuit shown in figure 4.13, which was designed by Petr Porazil from PiKRON company.

The resonance circuit is a simple series LC oscillator circuit resonating at 125 kHz. It is excited by PWM from nRF52833 pin isolated by non-inverting buffer 74AHC1G125. For the selected 345 µH antenna, the required capacitor value was calculated according to the resonance equation as follows:

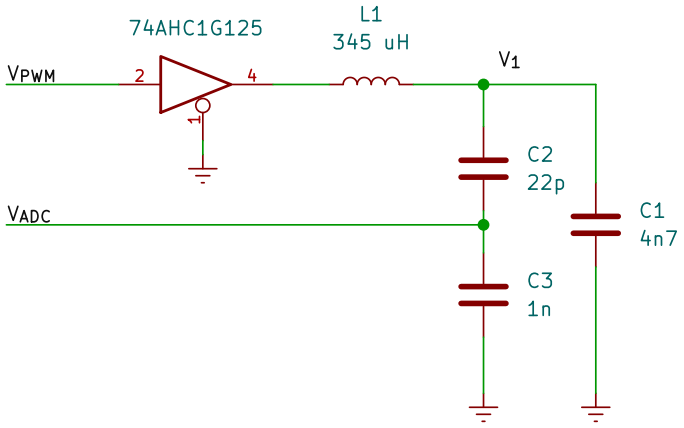


Figure 4.13: RFID resonance circuit

$$X_L = X_C \tag{4.6}$$

$$2\pi f L = \frac{1}{2\pi f C} \tag{4.7}$$

$$C = \frac{1}{4\pi^2 f^2 L}. \tag{4.8}$$

Therefore, the resulting capacitor C_1 value which was used is 4.7 nF.

Because the voltage V_1 on capacitor C_1 might be higher than the allowed ADC input range due to the resonance, a capacitive voltage divider was added.

$$V_{ADC} = \frac{X_{C_3}}{X_{C_2} + X_{C_3}} V_1 \tag{4.9}$$

$$V_{ADC} = \frac{C_2}{C_2 + C_3} V_1, \tag{4.10}$$

where X_{C_2} and X_{C_3} are capacitances of capacitors C_2 and C_3 respectively.

For the selected capacitor values $C_2 = 22\,\text{pF}$ and $C_3 = 1\,\text{nF}$, the overall impact of capacitive voltage divider on the LC resonance is negligible. Because of $C_3 >> C_2$, $C_{2,3} \approx C_2$ and because $C_1 >> C_2$, the overall capacitance $C_{1,2,3}$ is approximately equal to C_1.

The sampled voltage V_{ADC} on capacitor C_3 equals to $21.5 \times 10^{-3}\, V_1$.

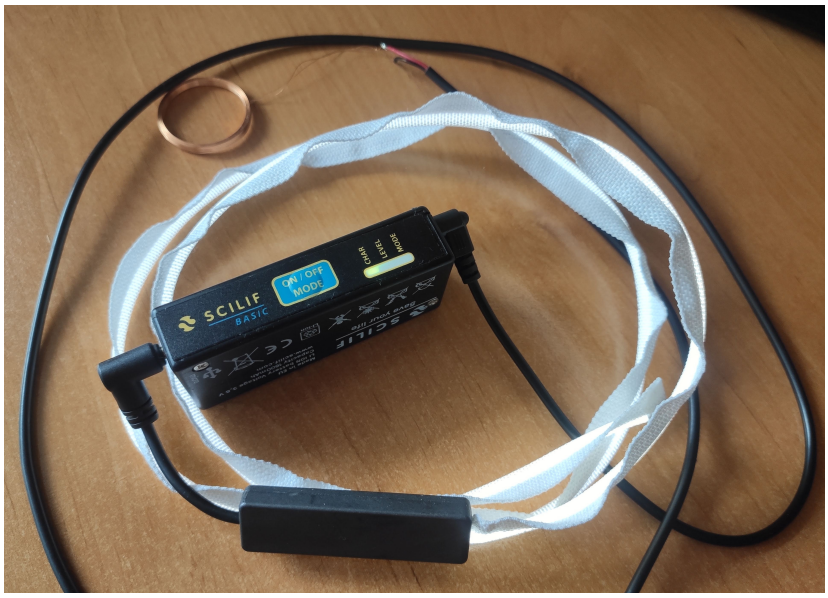


Figure 4.14: Light control unit with RFID reader and a light guide

Overall Design

The overall design of the light control unit accompanied with the RFID reader is shown in figure 4.14. The LC RFID reader was soldered on a small prototype board and connected instead of one RGB LED directly to the light control unit board. A secondary 2.5 mm jack was used to connect the coil externally.

4.3.3 EM4100 RFID Tag

The compatible tags are passive, low frequency, read-only RFID tags (see section 3.2.1 about their principles) in a key fob form based on a cheap EM4100 chip. The EM4100 is a CMOS integrated circuit powered by an external coil made by the company EM Microelectronic [4]. When the tag enters the near-field region created by an RFID reader, it draws power from the field by inductive coupling principle and it sends the data from a factory preprogrammed memory back to the reader by modulating the RF field. The 64 bits of data are read-only and represent a unique identifier of the tag, which is also printed on the PVC, so it can not be changed.
The RFID reader must know how the data is stored in the tag and which protocol to use to extract it - the demodulation and decoding method.

The main advantages of low-frequency EM4100 tags are the adoption of CMOS technology which makes them low-cost, the ability to penetrate water, organic tissue, etc. and a resistance to electromagnetic interference.
The main disadvantages are the the limited memory capacity, low data rate, low read range and compared with HF tags more complicated and expensive antenna.

Chapter 5

Software Development - Firmware

In this chapter, a comprehensive overview of software development tools and techniques is provided. The first sections 5.1 - 5.2 generally describe the process of development on nRF52833. The mutual aspects (BLE properties and application logic) of both the light control unit and floating button are analyzed in section 5.4 and 5.5. The following two sections focus on each device individually, showing their application state diagrams. In the last sections 5.8 - 5.10, a detailed description of pertinent system parts is given - such as button clicks debouncing, OTA upgrade or battery measurement.

5.1 Tools Used

The firmware was written in C language with the help of Nordic SDK version 17 [21]. The SDK includes various drivers, libraries, examples and most importantly, a SoftDevice, a BLE protocol stack provided as a precompiled and linked binary file [19]. The SoftDevice types differ in complexity and performance (for instance, the maximum number of concurrent links they can handle or the BLE role they support etc.). The light control unit and floating button are based on the SofDevice **S132** and **S140**, respectively.

Apart from SoftDevice, the HAL libraries (called NRFX), some additional middle-layers (RTC based timer/scheduler), utilities (logging module) and the BLE middle-layers were incorporated into the project. A crucial role plays, for instance, a peer-manager library which provides a whole infrastructure to security management and cryptography operations.

5.1.1 Programming

The programs are flashed to targets through OpenOCD (Open On-Chip Debugger), which uses a built-in SWD adapter on the microbit board and an external programmer on the light control unit board to communicate with the nRF52 chip.
OpenOCD complies with a remote gdbserver protocol, and therefore, gdb (GNU Debugger) with a graphical user interface DDD (Data Display Debugger) was used for applications debugging.

To flash all software parts at once (SoftDevice, application and bootloader - see section 5.2), a python script `merge_hex` was written that merges all three .hex files and converts them to a binary form.
An additional, python-based tool called `nrfutil` provided by Nordic directly allows generating specific bootloader settings and creating an initialization packet for DFU upgrade - see section 5.10.4 for a detailed description.

5.2 nRF52 Memory Layout

This section briefly introduces the memory layout of nRF52 and shows where all parts of device firmware (SoftDevice, application and bootloader) are located.

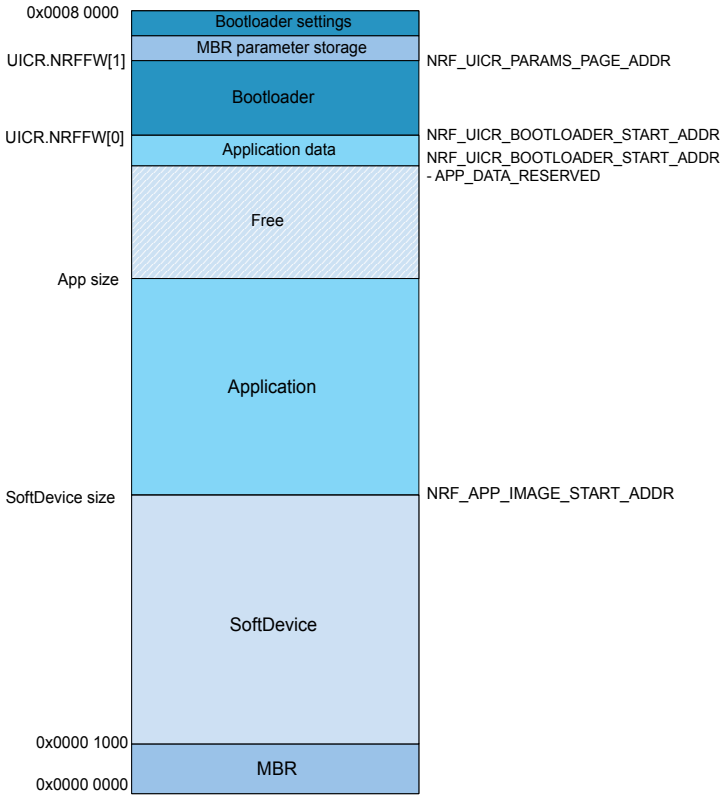


Figure 5.1: nRF52 memory regions (taken from [22])

MBR (Master Boot Record)
The MBR table is a part of the SoftDevice and it occupies 4 kB of memory at addresses 0x0000 - 0x1000. Its primary function is to provide an interface for DFU upgrades, ensuring its continuity in case of a power loss. It also contains the System Vector table, so all interrupts are at first processed by MBR and then forwarded to the bootloader or SofDevice [19].

SoftDevice
SoftDevice is in charge of the low-level BLE communication. The amount of occupied memory depends on the SoftDevice type. Each SoftDevice includes

a predefined hardcoded start address of the application where it jumps [19].

Application
The application address space starts from a predefined address determined by SoftDevice type - NRF_APP_IMAGE_START_ADDR and ends at the start of the bootloader - NRF_UICR_BOOTLOADER_START_ADDR [14].

Bootloader
A bootloader is a minimal piece of code responsible for booting into the application or entering DFU mode. While in DFU mode, it enables receiving and processing wireless firmware updates - see section 5.10 about OTA.
If the bootloader start address is present at UICR (User Information Configuration Register) in non-volatile memory, the MBR boots the bootloader instead of SoftDevice [14].

Bootloader Settings
A bootloader settings page occupies 4 kB of flash at address 0x7E000, and it contains information about the current DFU process and the metadata about the installed application, such as firmware and hardware version [14].

MBR Params Page
The last 4 kB of flash is dedicated to MBR parameters. It stores the state information of flash operations or DFU process, allowing their recovery [19].

5.3 Code Structure

This section introduces the code structure and practices that were followed during the development. Furthermore, a focus is given to a description of some essential code parts..

The Nordic SDK is separated from the source code and can be downloaded separately. The workspace maintained through git is structured into multiple folders, each one representing an independent project/executable program - light control unit application, floating button application and bootloaders.

The workspace structure:

- `Sunfibre_LightingUnit_CustomBoard`
- `Sunfibre_Bootloader_CustomBoard`
- `Sunfibre_FloatingButton_Microbit`
- `Sunfibre_Bootloader_Microbit`
- `shared`

The common parts of code shared among all projects are placed in a folder `shared`. It contains BSPs of light control unit board and microbit board, shared libraries (such as custom advertising library - `app_ble_advertising.c`, button library - `app_button.c` and the implementation of client and server part of BLE services).

5.3.1 Configuration Files

There are several configuration files in every project, such as:

`sdk_config.h`
This file specifies all configurations needed by Nordic SDK (SoftDevice, peripherals, logging . . .). It is assumed that these settings are fixed and won't change during development.

`app_sdk_config.h`
This file contains configurations needed by Nordic SDK too, but they might get changed during the development (logging level, debug mode, link counts etc.) or they are considered as important to point out (DFU entering method etc.).

`app_config.h`
The settings and macros used in application logic are declared here. For instance, light control unit LED colors corresponding to application states, the application timer intervals or battery level colors etc.

`app_ble_config.h`
All various application related BLE settings are listed in this file. It specifies the connection, security, advertising, or GAP parameters (device name, manufacturer information, etc.).

5.3.2 Application Logic Source Files

The application (business) logic is split from peripherals source code in a file called `businesslogic.c`, containing all user-related functionality and implementing the application state diagrams (shown in figures 5.5 - 5.8).
Application logic regarding BLE is implemented separately in `app_ble.c`. The BLE events are raised as interrupts by SoftDevice - GAP events (connection, disconnection. . .), security manager events (encryption started, bonding finished. . .), scanning events or database-discovery events. Some of these events are propagated to the business logic through application handlers, and appropriate actions are undertaken.

Additionally, the application-specific behaviour around advertising and GATT services is implemented in standalone files `app_ble_advertising.c` and `app_ble_services.c`.

5.4 BLE Properties and Modes of Operation

This section aims to describe the BLE configurations, modes of operation and parameters of both devices.
The additional, distinctive BLE properties of the light control unit and floating button are addressed in sections 5.6.1 and 5.7.1, respectively. The theoretical background of BLE is provided in section 3.1.

To begin with the modes of operation, the light control unit works as a GAP peripheral (slave & advertiser) and a GATT server. The floating button

can play the same role as the light control unit, but it is primarily intended to operate as a GAP central (master & scanner) and a GATT client.

5.4.1 Advertising and Scanning

Both, the light control unit and floating button implement the active scanning procedure (see section 3.1.2). The light control unit can only advertise, on the contrary, the floating button can behave as both, scanner and advertiser.

Advertising Data

The types and values of AD elements included in the advertising packet are shown in table 5.1. The structure of the whole advertising indication packet is shown in figure 5.2 (the AD element types are bold).

To explain some of them, the AD flags determine the discoverability and Bluetooth technology used, the appearance element define the external appearance of the device and the manufacturer data contains the name of the producer.
The scan response packet includes only the UUID of either Light Control Service or Monitor Service (see section 5.4.4).

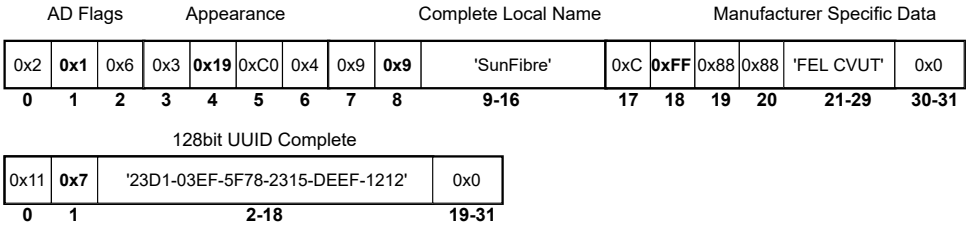


Figure 5.2: Advertising and scan response packet of light control unit

AD Type	Value
AD Flags **0x1**	0x6 - General Disc. Mode, BR/EDR Not Sup.
Appearance **0x19**	0xC004 - Generic Control Device
Complete Local Name **0x9**	SunFibre
Manufacturer Data **0xFF**	id: 0x8888, name: 'FEL CVUT'
128bit UUID Complete **0x7**	23D1-03EF-5F78-2315-DEEF-1212

Table 5.1: AD elements of the light control unit

Advertising Modes

In order to optimize the battery consumption, two different advertising modes were configured (fast and slow). The fast advertising runs for first three seconds after the advertising is turned on. In this mode, the advertising packets are broadcasted in short intervals to scan the peripheral quickly. If no device connects, a slow advertising mode is turned on when the packets are

transmitted in longer intervals to save the battery (see table with parameters 5.2).

The device stays advertising until a connection occurs or until a time interval determined by the current application state expires (see the section about application timer - 5.5.3).

Scanning Mode

The floating button listens for advertising packets in 2500 ms intervals with an active scanning window of 1250 ms, meaning the radio is turned on 50 % of the scanning time. Hence, the worst case latency of successfully scanning the light control unit in a slow advertising mode is 2000 ms. See section 3.1.2 which defines these parameters.

The scanned advertising packets are filtered according to the device name. Once the peripheral is connected or when the application timer elapses, the scanning is stopped.

5.4.2 Connection Parameters

The initial connection parameters are configured by central (floating button or smartphone). They allow greater throughput so that the data can be exchanged quickly at the start (initial reading of characteristics, writing to CCCD descriptors etc.). After five seconds from the connection establishment, the light control unit sends a `Connection Parameter Update Request` demanding a larger connection interval in order to reduce the battery consumption.

The value of the connection interval was set as a trade-off between battery consumption and a reasonable latency, which is important especially when the light mode gets changed. The slave latency is set to zero because the read or write commands initiated by central prevails.

See section 3.1.2 which defines these parameters and their values in table 5.2.

Throughput Parameters

Due to the fact that the data length of all BLE characteristics does not exceed 4 bytes (see section 5.4.4) and very few packet exchanges are needed, all throughput parameters were set to the lowest possible values to save the battery.

5.4.3 Security

Because both, the light control unit and floating button are keyboardless and screenless devices, displaying a passkey is not possible, so the only possible STK generation and exchange method is **Just Works**. Hence, security mode 1, level 2 (encryption required, authentication and authorization not required) is necessary for communication and data exchanges.

Type	Value
Fast Advertising Interval	100 ms
Fast Advertising Duration	3000 ms
Slow Advertising Interval	1000 ms
Slow Advertising Duration	x
Scanning Interval	2500 ms
Scanning Active Window	1250 ms
Scanning Duration	x
Initial Connection Interval	100 ms
Connection Interval	250 ms
Connection Slave Latency	0
Connection Supervision Timeout	4000 ms
MTU Size	23
Data Length	27
Event Length	7.5 ms
TX Power Level	0 dBm

Table 5.2: Selected BLE parameters

This security level doesn't provide any protection against man-in-the-middle (MITM) attacks. However, it is assumed that the users will pair their devices in a safe environment where the probability of MITM attacks is very low. After that, the LTKs are always used for the link encryption.

See section 3.1.3 introducing the terms used here.

5.4.4 BLE Services and Characteristics

In total, three custom primary services with 128-bit UUID were implemented to provide interface to manipulate with data - *Light Control Service*, *Monitor Service* and *RFID Service*. The light control unit uses all of them, the floating button only the *Monitor Service* when in peripheral mode.
Additionally, *Secure DFU Service* service was added to both devices. It is also a primary service with 128-bit UUID implemented by Nordic.

See section 3.1.4 about GATT data hierarchy.

LED Control Service

Base UUID: **1212-efde-1523-785fef13d123**

The LED Control Service serves as a controller allowing changing the light modes (see section 2.1) and as a monitor providing LED current and voltage. In total, it contains three characteristics.

Light Mode Characteristic

The characteristic value is a one-byte long integer that represents the current light mode. The value corresponds with the light modes in section 2.3.
A light mode can be either incremented by writing -1 to the characteristic or changed directly by writing the light mode number directly. The characteristic

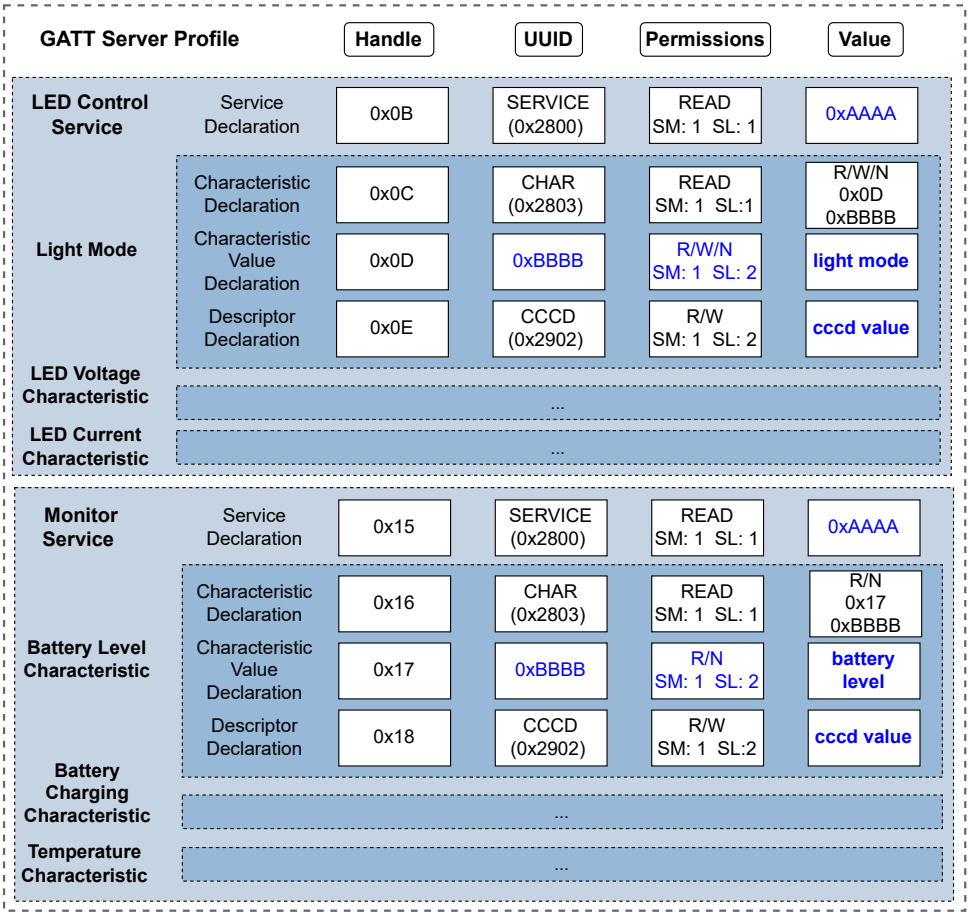


Figure 5.3: Section of GATT attribute table - LED Control and Monitor services

value is also updated after every light mode change that does not come from BLE - button press or RFID detection.
In case of a light mode change, a notification is sent to all connected clients if enabled previously. The characteristic properties are read/write/notify. The link must be encrypted to read or write - security mode 1, security level 2.

LED Voltage & LED Current Characteristics

Both characteristic are two byte long signed integers representing LED voltage in millivolts and LED current in milliamps. They get updated after ADC measurement alongside with Battery Level Characteristic. They can only be read and notifications could be sent if their values change.

Monitor Service

Base UUID: **1413-f0df-1624-7960f014d224**

The Monitor Service exposes additional monitored data to a client. It includes these three characteristics:

- **Battery Level Characteristic**,

- **Battery Charging Characteristic**,
- **Temperature Characteristic**.

The Battery Level and Battery Charging characteristic values are one-byte long and represent the estimated battery level and charging status. The Temperature characteristic value is a four-byte long characteristic and reflects the board's temperature in Celsius degrees.
All three characteristic values can only be read. Section 5.9.5 explains how and how often they are updated. A notification for a particular characteristic is sent to each connected client only if its value changes and when notifications are enabled. The characteristic properties are read/notify. The security requirements to obtain data are the same as for Light Mode Characteristic.

RFID Service

Base UUID: **1514-f1e0-1725-7A61f115d325**

The RFID Service is in charge of controlling RFID functionality. It is made up of three characteristics too.

RFID Enabled Characteristic

This characteristic value is one byte long - either logic one or zero and determines whether the RFID detection is turned on or off. The characteristic properties are read/write.

RFID Paired Tag ID Characteristic

Whenever the user wants to change the tag that switches the light modes, he needs to update this characteristic. It is read/write characteristic and its value represents unsigned 32-bit integer (tag id).

RFID Detected Tag ID Characteristic

Regardless of which tag is paired (set to increment the light mode), this characteristic always contains the tag id that has been detected the last time. The characteristic value is also 4 byte long unsigned integer and its properties are read/notify.

The security requirements for obtaining data from RFID Service characteristics are the same as for previous services.

Secure DFU Service

Base UUID: **f315-4f60-9fB8-838830daea50**

The main purpose of the Secure DFU Service is a remote (buttonless) activation of the DFU mode.

Buttonless DFU Characteristic

When a client writes to this characteristic, a unique value is written to the GPREGRET register, and a system reset to DFU mode is executed (see section 5.10.3 which comments the DFU activation process). The characteristic properties are write/indicate. Neither the encryption of the link nor the

authorization is required - security mode 1, security level 1 (see section 3.1.3 explaining the security modes).

Service and Characteristic Discovery

To read or write a characteristic value or a descriptor, the central must know its attribute handle.
However, because the handles are assigned to the attributes by SoftDevice and might have arbitrary values, the service and characteristics discovery is necessary. At first, the central performs a discovery of primary services followed by a discovery of all characteristics of a particular service. The characteristic value attribute handles are read from characteristic declarations - see the model of GATT server in figure 5.3 and section 3.1.4.

Since the attribute handles are consistent, if the layout of the GATT server model does not change, they are cached and stored in flash so that during the next connection, the discovery procedure is not needed.

Nordic libraries were used for discovery procedures and attribute handles caching on the floating button.

5.5 Application Logic

A general explanation of system devices' operation and communication between them is described in this section. It further introduces common terms and behaviour used in the application logic. The more detailed description of the light control unit and floating button is further presented in sections 5.6 and 5.7.

In general, the light control unit, a BLE peripheral, can be connected to multiple centrals (floating buttons or smartphones/smartwatches) at the same time. To connect a new central, the bonding procedure must be carried out first. The bonding is required only during the first connection. Later, it can be skipped and only connection procedure is needed.

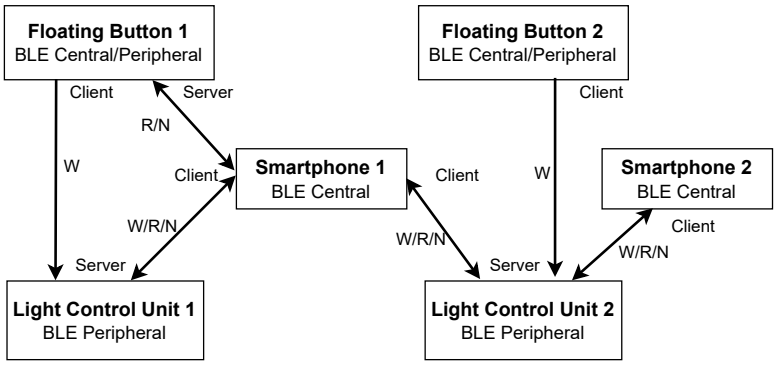


Figure 5.4: Visualization of possible communication topology: *R/W/N* stand for GATT read write or notify procedures

See section 5.6.3 with detailed description of these procedures. When a

central is connected to the light control unit, it can read its BLE characteristics (light mode, monitored data and RFID data) or write to them - change the light mode, RFID properties or enter the DFU mode.
Notifications synchronise these characteristic values with centrals. So, for example, if the light mode is changed by RFID, its new value is sent to all connected centrals.

The floating button can operate as both, BLE central and BLE peripheral, simultaneously. Changing the light modes on the light control unit is possible in the central role. The peripheral mode enables reading the floating button monitored data (battery level, charging status, temperature) through a smartphone application.

5.5.1 Application States

The application states prescribe the system's behaviour (whether the advertising/scanning is turned on, the whitelist is used or when the application timer expires etc.). The system actions (procedures) are triggered by application events (section 5.5.2) and are dependent on the current application state.
The application states differ for light control unit and floating button - see the paragraphs about them in sections 5.6.3 and 5.7.3.

Application Event	Source
EVT_BTN_SHORT_CLICK	button
EVT_BTN_MEDIUM_CLICK	button
EVT_BTN_LONG_CLICK	button
EVT_BTN_VERY_LONG_CLICK	button
EVT_BTN_PRESS_START	button
EVT_BTN_PRESS_INTERVALPASSED	button
EVT_BLE_CONNECTED	BLE
EVT_BLE_CONNECTED_AND_BONDED	BLE
EVT_BLE_DISCONNECTED	BLE
EVT_BLE_DISCONNECTED	BLE
EVT_BLE_SECURITY_START	BLE
EVT_BLE_SECURITY_ERROR	BLE
EVT_BLE_BONDS_REMOVED	BLE
EVT_BATTERY_CHARGER_CONNECTED	battery
EVT_BATTERY_CHARGER_DISCONNECTED	battery
EVT_BATTERY_MEASURED	battery
EVT_RFID_DETECTED	RFID
EVT_TIMER	RTC

Table 5.3: A list of supported application events

5.5.2 Application Events

The application events are generated by various system parts such as BLE, battery, buttons, RFID or application timer. They are handled in an application handler based on the type of the event and the current application state. Some events require additional context - further specification of the event, for example, which connection was disconnected.
The table 5.3 shows the list of all application events. Their meaning should be self-explanatory, and their names correspond to the labels of dashed lines in the state diagrams (figures 5.5 - 5.8 and 5.9, 5.10).

5.5.3 Application Timer

The application timer is based on RTC. As long as it elapses, an interrupt is triggered, application event EVT_TIMER is generated and the application enters a new state. Depending on the new state, the value of the timer is set (see table 5.4 or 5.6 below for application timer values).
In order to prevent the timer from firing during long clicks, a detection of a button press (application event EVT_BTN_PRESS_START) immediately resets/extends the current timer regardless of the current state.

5.6 Light Control Unit

This section provides a specification and detailed description of BLE properties, user interface and the application logic related to the light control unit. The general description with functional requirements is provided in section 2.1. The required user interactions with the light control unit are specified in section 2.6.

5.6.1 BLE Properties

According to GAP, the light control unit behaves as a scannable and connectable peripheral (slave). In particular application states (see 5.6.3), it advertises with the name **SunFibre** and accepts incoming connections from a centrals.
When the connection is established, the unit follows the central's timings (connection interval) but requests an update of connection parameters after a few seconds.
The data are read or written regularly through BLE services. Up to 5 centrals could be connected at the same time.

More information on advertising, connection parameters and BLE services can be found in section 5.4. The structure of advertising packet is in figure 5.2 and a part of GATT attribute table in figure 5.3.

5.6.2 User Interface

One push-button and three RGB LEDs on the light control unit board were used to design a user interface.

Button

Pressing the button generates button application events (see 5.5.2). A short, medium, long and very long clicks are distinguished (see the section 5.8 commenting button click handling and table 5.7 with click's durations).
To put it simply, a short button click changes the light mode, a medium-long click makes the device visible for known, already bonded centrals (joining procedure), a long click starts searching for new devices (bonding procedure), and a very long click resets the device.

RGB LEDs

The first LED is called *state LED* and its color and blinking determine the application state (see section 5.5.1).
The second one, *battery LED*, indicates the level of battery (see the colors corresponding to battery levels in table 5.9. If the LED is blinking, the battery is being charged.
The third, *helper LED*, blinks if an interval for recognizing a particularly long click passed (application event EVT_BTN_PRESS_INTERVALPASSED. As a result, the number of blinks indicates the user when to release the button for a particular long click.

5.6.3 Application Logic

This subsection introduces the application logic of the light control unit that has been designed and implemented according to customer requirements. The functionality is split into multiple so-called procedures, each one further explained by a state diagram.

Application State	Description	Timer	LED
SYSTEM OFF	deep sleep	∞	off
IDLE	sleep, action waiting	20 s	off
NOT CONNECTED	$C = 0$, action waiting	20 s	🔴
CONNECTED	$C > 0$, action waiting	∞	🟢
JOINING ADVERTISING	$C = 0$, connection waiting, whitelist	40 s	B. 🟡
	$C > 0$, connection waiting, whitelist	15 s	B. 🟢
JOINING CONNECTED	$C = 0$, security request waiting	10 s	🟡
	$C > 0$, security request waiting	10 s	🟢
BONDING ADVERTISING	connection waiting, no whitelist	15 s	B. 🔵
BONDING CONNECTED	pairing request waiting	10 s	🔵

Table 5.4: Application states of light control unit - C denotes the number of previously connected centrals, *B.* means blinking

Application States

The application states on the light control unit are distinguished by color of *state LED*. If the device is advertising, the LED is blinking. Immediately after a connection, the blinking is stopped.

The table 5.4 briefly explains the behaviour in application states and shows their corresponding LED color and application timer interval.

Wakeup Procedure

IDLE is the default state that the application enters after reboot. To proceed to the joining procedure, the user must click on the button (medium-long). In case that there are no previously bonded peers saved in flash, the device moves to **NOT CONNECTED** state, turning the *state LED* red and allowing further only bonding procedure.

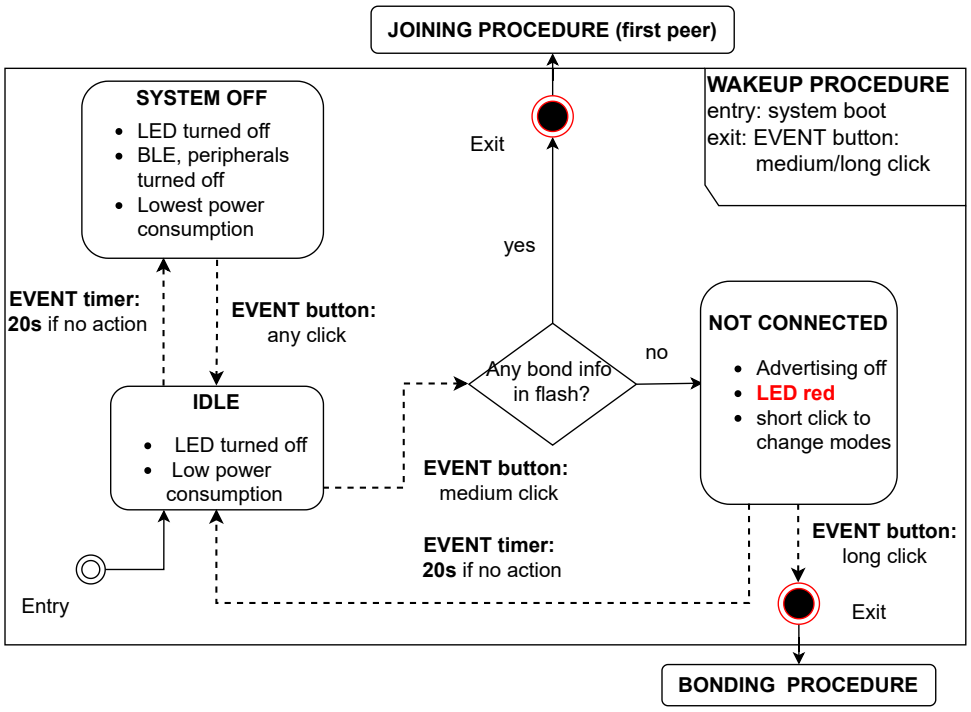


Figure 5.5: State diagram - application start (wakeup)

Deep Sleep

If the battery level is under a critical level (see 5.9.5) or when the inactivity timer elapses in IDLE state, the unit enters **SYSTEM OFF** state. In order to reduce the battery usage to the minimum, all peripherals besides GPIO are turned off and the device wakes up only by an arbitrary long button click.

Connected Procedure

When in **CONNECTED** state, the device is connected to at least one central and does not advertise. It allows R/W operations of BLE characteristics (see

section 5.4.4).
By a long button click, the user can start the bonding procedure with a new central, a medium-long button click allows searching for bonded centrals - joining procedure. If all connected devices disconnect, JOINING ADVERTISING state is entered automatically.

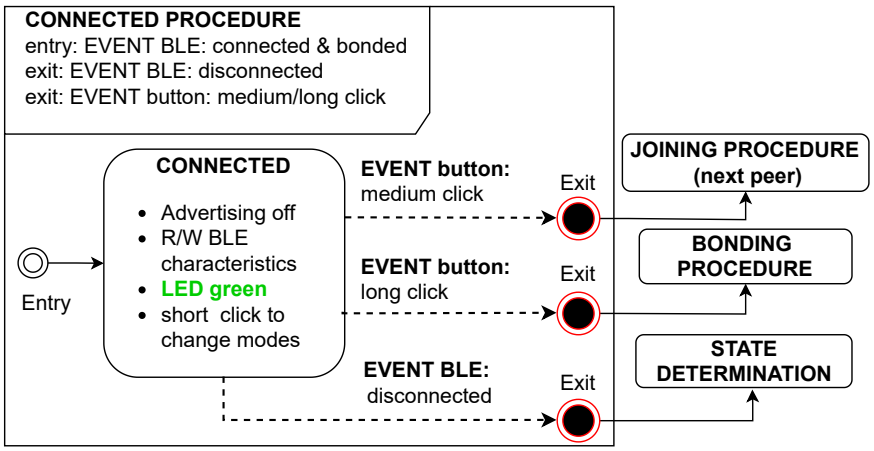


Figure 5.6: State diagram - connected

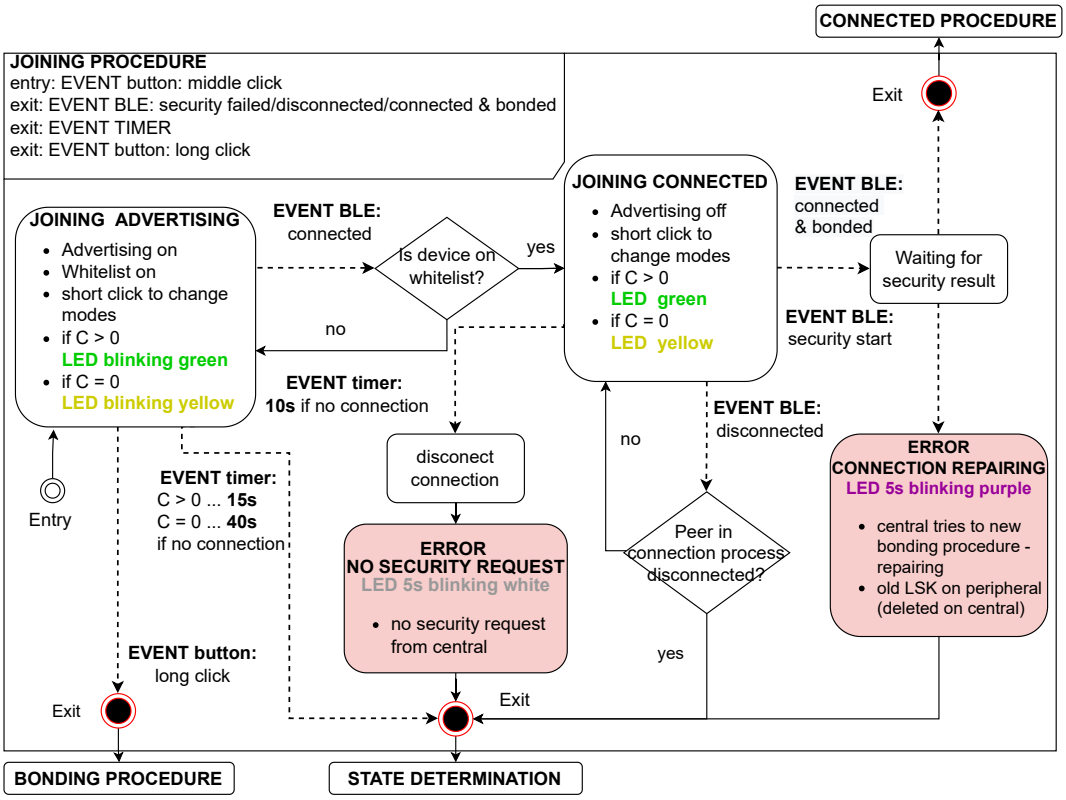


Figure 5.7: State diagram - joining

Joining Procedure

The joining procedure can be started by a medium-long click from states: IDLE or CONNECTED. While in **JOINING ADVERTISING** state, only previously bonded devices are allowed to connect (the light control unit advertises with a whitelist). Thus all connection requests from unknown centrals are rejected. The *state LED* is blinking, and its color is either yellow if no centrals are connected, or green if at least one another central is connected).
If a connection from a known central occurs, advertising is stopped and LED stops blinking - **JOINING CONNECTED**. The device waits until the connected central starts the authentication - GAP encryption procedure which starts the encryption of the link with the previously negotiated LTK security key (see section 3.1.3).
Providing that the encryption procedure is successful, the CONNECTED state is entered. Otherwise, an error is signalized (discussed in the paragraph about error states).

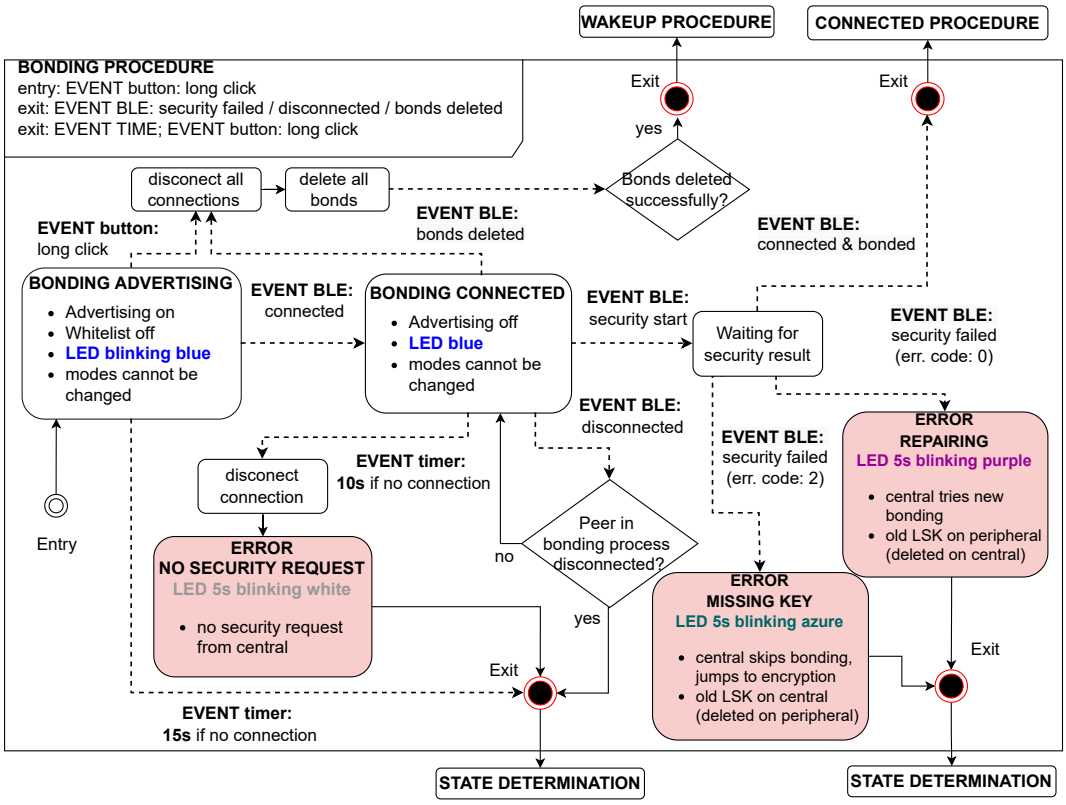


Figure 5.8: State diagram - bonding

Bonding Procedure

The bonding procedure is entered by a long button click from any of the states: NOT CONNECTED, JOINING ADVERTISING or CONNECTED. During **BONDING ADVERTISING** state, all connection requests are accepted (the device advertises without a whitelist).

In case of a new connection, the advertising is stopped, and the *state LED* stops blinking - **BONDING CONNECTED**. The device waits until the central initiates the GAP authentication procedure (pairing and bonding procedures - exchange of security keys, see section 3.1.3).
In case of success, the CONNECTED state is entered. Otherwise, an error is indicated (further described in paragraph about error states).

Deleting Bonds

The user can delete all stored long term security keys (LTKs) from flash. This action requires two long button clicks. Upon the removal, the device enters NOT CONNECTED state.

Error States

Bonding or joining procedures might fail for several reasons. The most important ones are indicated to the user by a few *state LED* blinks and are shown in the table 5.5. The color of the LED determines the type of the error.

NO SECURITY REQUEST refers to a situation when a central connects to the peripheral but does start the GAP authentication or encryption procedure so that the application timer elapses.
MISSING KEY error might occur if central tries to encrypt the connection by using a LTK that is not present or valid on the peripheral. This error is usually caused by deleting bonding information on the peripheral but not on the central.
REPAIRING appears when a central initiates the authentication procedure (pairing), and generates a new STK despite being previously bonded (the old LTK is still stored on the peripheral). This error is usually caused by the deletion of bonding information on the central but not on the peripheral.
Nevertheless, this error might be prevented when repairing is explicitly allowed. Then the central can always bond with the light control unit regardless of previously saved LTK.

Error	Reason	LED
NO SECURITY REQUEST	no encryption/authentication procedure	○
MISSING KEY	no LTK on peripheral, old LTK on central	●
REPAIRING	old LTK on peripheral, new LTK on central	●

Table 5.5: Error states of light control unit.

5.7 Floating Button

This section is devoted to the specification of BLE properties and the user interface of the floating button. It also describes in detail the application logic and shows the application state diagrams. The general description with functional requirements of the floating button is provided in the section 2.2. The required user interactions with floating button are specified in section 2.6.

5.7.1 BLE Properties

In comparison with the light control unit, the floating button can operate as both central and peripheral.
In the central (master) mode, it behaves as an active scanner, listening for advertising packets and initiating a connection with the first found light control unit. It determines the initial connection parameters and performs a discovery procedure of BLE characteristics. When the connection is established and discovery completed, it can write or read to *Light Control Service* characteristics on the light control unit.
At the same time, the floating button can act as a scannable and connectable peripheral (slave), sending connectable advertising packets with name **SunFibre-btn** and accepting incoming connections from a central - smartphone. The central then determines the connection intervals and can read characteristic values of *Monitor Service* or write to the *Secure DFU Service*.
Only one connection is possible in both modes (one central and one peripheral).

More information on advertising, connection parameters and BLE services can be found in section 5.4.

5.7.2 User Interface

The floating button's user interface is based on two-push buttons and a 5x5 LED array.

Button

Clicking on the buttons generates application events (5.5.2). A short, medium, long and very long clicks are distinguished (see table 5.7) in the same way as on the light control unit.
The left button controls the central mode. A short click allows changing the light modes on the peripheral (only if connected), a medium-long click makes the device scan for peripherals, a long cancels the peripheral mode, and a very long click removes bonds.
The peripheral mode is controlled by the second (right button). To put it simply, a medium click starts advertising, a long click allows disconnecting a connected device and a very long click resets the device.

LED Matrix

The LED matrix helps to determine the application state by showing a particular letter (see table 5.6). A control module was implemented in order to allow drawing a bitmap on the matrix. A timer refreshes the rows with 200 Hz frequency.

5.7.3 Application Logic

In this subsection, the application logic of the floating button is introduced and its behaviour explained on state diagrams.

Application States

The floating button keeps two application states - one for central mode and one for peripheral mode. The application events generated by the left button influence the central state, the events from the right button the peripheral state. The BLE application events are handled based on the type of connection, whether they apply to central or peripheral mode.

Application State	Description	Timer	LED Matrix
SYSTEM OFF	deep sleep	∞	off
CENTRAL IDLE	action waiting	20 s	**I** iff P. IDLE
CENTRAL SCANNING	scanning, connection waiting	20 s	**S** iff P. IDLE
CENTRAL CONNECTED	action waiting	∞	**C** iff P. IDLE
CENTRAL ERROR	action waiting	∞	**E** iff P. IDLE
PERIPHERAL IDLE	action waiting	20 s	x
PERIPHERAL ADVERTISING	advertising, connection waiting	20 s	**P**
PERIPHERAL CONN. ACCEPTED	security request waiting	10 s	**P**
PERIPHERAL CONNECTED	action waiting	∞	**P**

Table 5.6: Application states of the floating button.

The application states are distinguished by the alphabet letters visualized by the LED matrix. Since the peripheral mode is less frequent and short-term, the peripheral states are prioritized. The central states (CENTRAL IDLE, CENTRAL SCANNING, CENTRAL CONNECTED) are visualized only if the peripheral state is IDLE. The letter **P** is used for all peripheral states, because the connection status could be immediately guessed from the smartphone application (the floating button can connect only to a smartphone in the peripheral mode).

The table 5.6 briefly describes the behaviour in particular application states. It also shows the application timer intervals and LED matrix letters associated with the states.

Wakeup Procedure

After the system reboot, the device enters central and peripheral IDLE states and waits for an action - start of scanning or advertising. The device might enter SYSTEM OFF on the same conditions as the light control unit.

Central - Scanning procedure

The scanning procedure is started from CENTRAL IDLE state by the left button medium-long click. In CENTRAL SCANNING state the device scans for advertising packets, filters them based on the target name. Once a light control unit is scanned, it is immediately connected and the GATT discovery procedure is started if the device attribute handles are not saved in flash. Subsequently, the link is either encrypted if a LTK security key exists on central or the authentication procedure is started (pairing and bonding). Providing that no security error occurs, the device stops scanning, and the CENTRAL CONNECTED state is entered. Otherwise, an error is indicated by the LED matrix showing the letter **E**. When successfully connected, the device can write the *LED Control Service* characteristics on the light control unit (see section 5.4.4).

Disconnection & Deleting Bonds

In the CENTRAL CONNECTED state, the user is able to disconnect the current connection by a long left button click. A very long left button click deletes all bonds from flash.

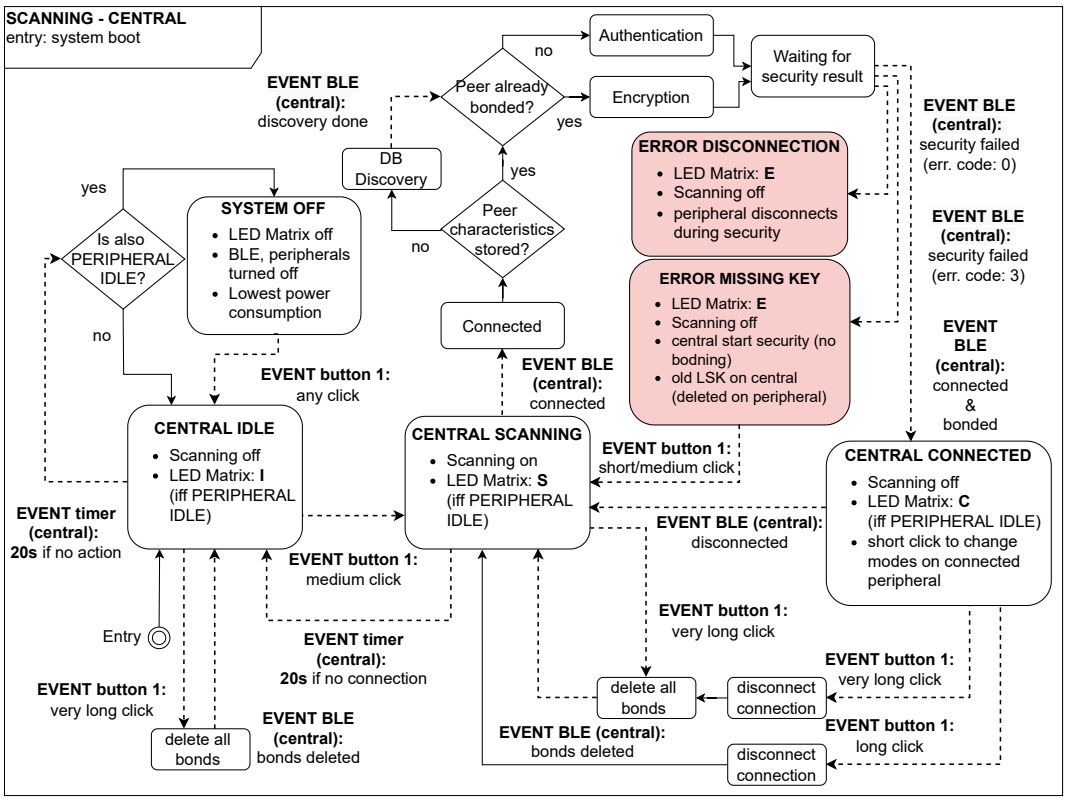


Figure 5.9: State diagram - scanning (central)

Advertising - Peripheral Procedure

The PERIPHERAL IDLE state is the default application state for peripheral mode. A medium-long click on the right button activates the advertising. The peripheral mode is then visualized by letter **P** on the LED matrix regardless of the current central state. Within 20 seconds, a central must connect and secure the link. While in PERIPHERAL CONNECTED state, the characteristics from *Monitor Service* and *Secure DFU Service* can be read or written.

Disconnection & Rebooting

The long right button click quits the peripheral mode - disconnects the current connection in case of PERIPHERAL CONNECTED state and stops advertising in PERIPHERAL ADVERTISING state. In an arbitrary state, a very long right button click reboots the device.

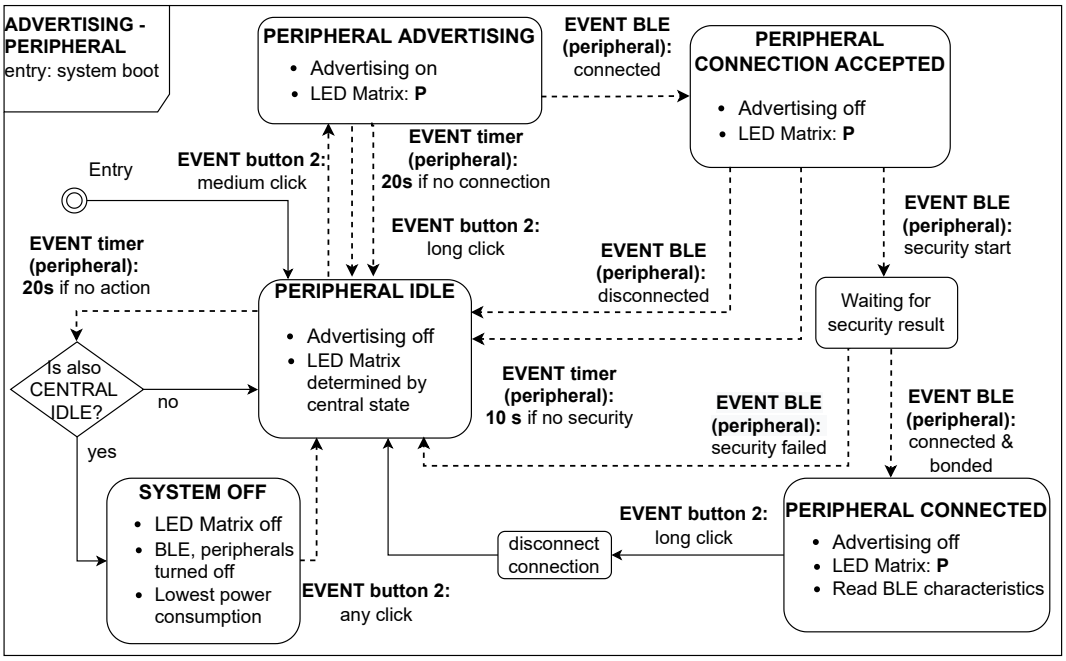


Figure 5.10: State diagram - advertising (peripheral)

5.8 Button Debouncing & Press Detection

This section aims at describing the software techniques implemented to prevent button bounces and provide press duration detection.

5.8.1 Button Debouncing

As with most mechanical switches, the basic push buttons on both boards suffer from the bouncing problem. Whenever two mechanical contacts in a button connect, they tend to bounce off each other a few times before settling down. Bouncing happens in a matter of milliseconds [34]. If the GPIO interrupts are used for detection, the MCU is fast enough to register all these oscillations and keeps calling the interrupt handler, which might overload the MCU.

An algorithm preventing bouncing uses a timer and GPIO interrupt. When an edge on the button GPIO pin occurs, three consecutive scans of the button state in 15 ms intervals are planned. The button state change is confirmed only if the pin state is the same in all three scans.

In the **GPIO ISR**, the further GPIO interrupts on this pin are disabled, and a timer compare event on channel one is scheduled to scan the button state (see figure 5.11 showing the algorithm).
Every time the **timer ISR** is called, the button state is checked at first. If the logical level of the pin is the same as the saved button state, the change is rejected, and the GPIO interrupt is enabled again. In the case of a successful third scan, the button state is changed, and the GPIO interrupt re-enabled.

One of the problems of this approach that is needed to take into account is a potential undetected button change after the read of the button state and before the GPIO interrupt is enabled. Nevertheless, this problem should be prevented by a latch register on nRF52 GPIO pins that holds the detected value, and when the interrupt is re-enabled, it immediately causes IRQ.

5.8.2 Button Application Events

As long as the button changes its state to pressed (active), the current timer value is saved, and the application event EVT_BTN_PRESS_START is generated. Subsequently, the timer compare event on channel two is scheduled for the end of the current click interval (1000ms for a short click). Once this interrupt fires, the application event EVT_BTN_PRESS_INTERVALPASSED is outputted, and a next event scheduled.
See the subsections 5.5.3 and 5.6.2 describing how are these events handled.
After the button state change to released (inactive), the duration of the click is calculated based on the current timer value and the saved start time of the press. The table 5.7 shows the duration for particularly long clicks and the application event that is triggered.

The logic described above is implemented in a library `app_button.c`. At most, three buttons are supported (the nRF52 timer has 6 compare events).

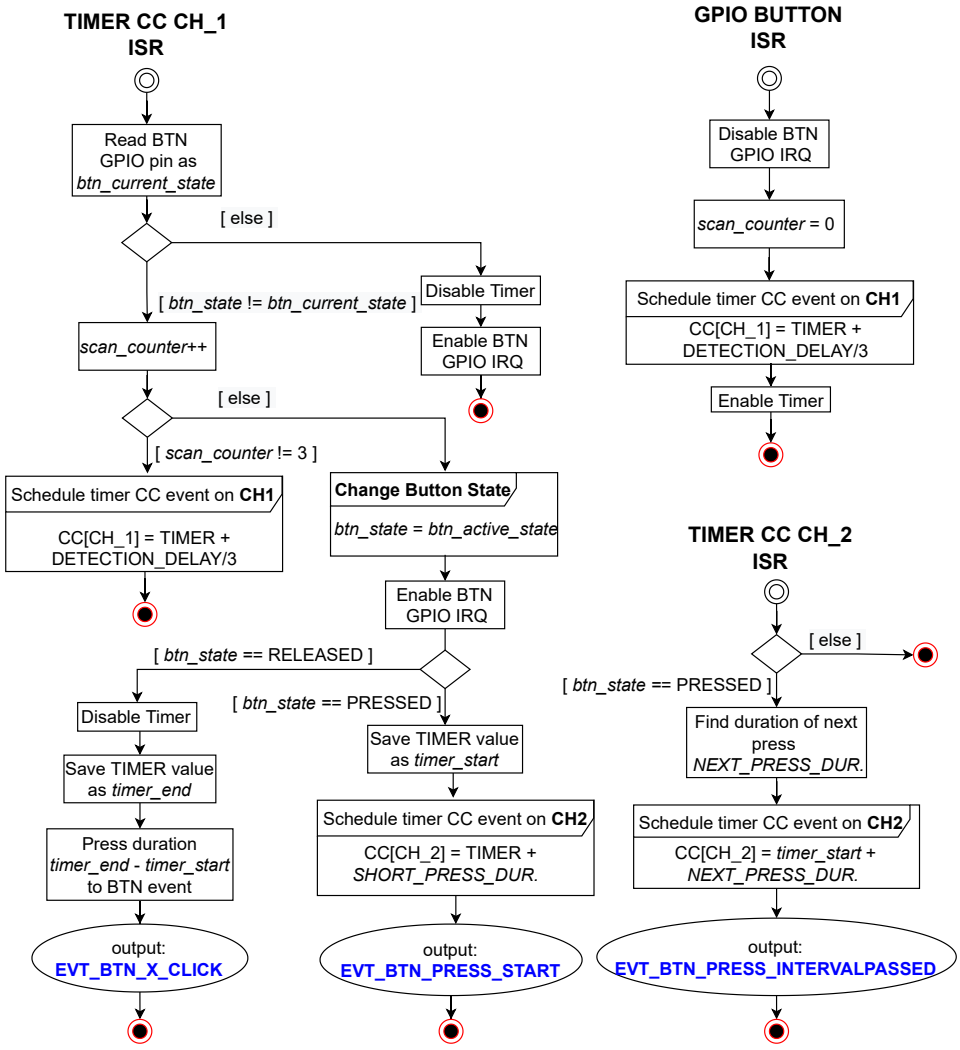


Figure 5.11: Button GPIO and timer interrupt handlers (ISRs)

Application Event	Press duration
EVT_BTN_SHORT_CLICK	45ms - 1s
EVT_BTN_MEDIUM_CLICK	1s - 3.5s
EVT_BTN_LONG_CLICK	3.5s - 8s
EVT_BTN_VERY_LONG_CLICK	8s - 20s

Table 5.7: Button click application events and their defaul duration

5.9 Battery, LED and Temperature Measurement

This section analyzes the light control unit's measurement of monitored variables and ADC sampling configuration. The focus is also given to the estimation of battery level and the overall incorporation into the application logic.

5.9.1 Battery Measurement

The battery measurement circuit is analyzed in the section 4.1.3. This subsection adds the ADC setup and subsection 5.9.3 further explains how the battery voltage is sampled.

For the channel configuration, the internal 0.6 V ADC reference was selected. In accordance with the maximum possible ADC input voltage (calculated in 4.2), the gain 1/3 was used enabling the input voltage to go up to 1.8 V. This gives the effective ADC range for battery measurement between 490 and 763 LSBs with sensitivity 1.758 mV.

5.9.2 LED Voltage and Current Measurement

The section 4.1.4 explains how the current flowing through the LED is controlled. However, the monitoring of LED state is still important task and therefore, it is sampled regularly, together with the battery.

The configuration of channels is the same as in case of battery channel apart from used ADC gains. For current measuring, the gain is 1 allowing maximum input 600 mV and for voltage 1/6 allowing the maximum input 3.6 V. Both values were adjusted according to the circuit parameters.

5.9.3 Sampling

The ADC samples all three channels (battery voltage, LED voltage and LED current) together in one sample task. The application timer based on RTC schedules sampling once per three seconds. If the required ADC channels are not initialized, for example because RFID configuration was used previously, the ADC needs to be reinitialized and channels configured.

Once the sampling is set up, five samples from each channel are taken. Each sample task is started by PPI (4.1.1) triggered by timer independently of the MCU. Same peripherals as for RFID sampling are used - see section 6.3.1.

When the sampling is done, an average is calculated and the BLE characteristics are updated.

Resolution	10 bit
Acquisition Time t_{acq}	3 µs
Reference	internal 0.6 V
Gain (battery)	1/3
Gain (LED voltage)	1/6
Gain (LED current)	1

Table 5.8: ADC channel parameters

5.9.4 Temperature Measurement

The temperature sensor is integrated directly on the nRF52 chip. The temperature range is greater than the allowed operating temperature range of the chip and the resolution is 0.25 degree [24, p. 425]. The temperature is measured with the same period as battery.

5.9.5 Battery Application Logic

When the sampling and averaging is done, the battery level is estimated and the indication is set.

Battery Level Estimation
Estimating the precise battery level is a complicated task. The battery discharge curve depends on the type of battery, temperature, discharge rate etc. For the sake of simplicity and according to the customer requirements, the battery levels were divided into five categories (see table 5.9). Once the battery level is estimated, the application event EVT_BATTERY_MEASURED is generated (see section 5.5.2 for all application events).

Level 5	4.0 V - 4.2 V	
Level 4	3.8 V - 4.0 V	
Level 3	3.6 V - 3.8 V	
Level 2	3.4 V - 3.6 V	
Level 1	below 3.4 V	off

Table 5.9: Battery levels and indication colors (on the light control unit)

On the light control unit, a particular battery level is indicated by using different colors of *battery LED* (see 5.6.2). On the floating button, the battery level can be obtained only through a mobile application.

The LED colors corresponding to the battery levels and the frequency of the measurements is configured in `app_config.h`.

Battery Charging Detection
Detecting whether the battery is being charged is done through GPIO interrupt for the STAT pin of the MCP73832 chip (see section 4.1.5). The logical level of the pin is read in the interrupt handler, and a battery application event is generated (such as EVT_BATTERY_CHARGER_CONNECTED - see 5.5.2).
The application event handler then updates the BLE characteristics, notifies all connected clients and sets the indication - the *battery LED* is blinking if the battery is being charged.

Battery Undervoltage Protection
When the estimated battery level is below 3.4 V, the firmware automatically enters the SYSTEM OFF mode in order to prevent the undercharging and further damaging of the battery. In this mode, the battery consumption is reduced to a minimum - all peripherals besides GPIO are turned off.

5.10 OTA DFU

OTA DFU stands for Over the Air Device Firmware Upgrade referring to the fact that the DFU package with new firmware is sent to the target device wirelessly over BLE. This section describes and explains this process and introduces its various parts.

5.10.1 Bootloader

The DFU process relies on the existence of a bootloader. The location of a bootloader and bootloader settings in memory is provided in the section 5.2. The bootloader is a minimal piece of code compiled and loaded onto the target separately from the main application. It is responsible for launching the application, entering the DFU mode or activating a new firmware [14].

Two bootloaders, one for the light control unit and one for the floating button, were implemented each one as a standalone project (see section 5.3 showing the code structure). Apart from using the different indications of DFU process steps, they are identical. The bootloaders implementation is based on nRF Secure Bootloader which uses security measures to protect the DFU process against malicious attackers accepting only DFU packages that are cryptographically signed with the correct key [18].

5.10.2 Bootup Validation

The bootloader checks the validity of the application every time during bootup. The CRC validation method (CRC32) is used for this purpose ensuring the data integrity of the application.
The application is searched at a first bank, predefined memory address determined by the SoftDevice type (see section 5.2). Although the DFU uses dual bank update, the new firmware is always copied to the first bank after a completed and validated upgrade (check the data transfer paragraph in section 5.10.4).

5.10.3 Buttonless DFU Activation

The DFU mode is activated by the bootloader if one of the following occurs:

- No valid application during bootup (see 5.10.2)
- A special value present in the GPREGRET register.

The GPREGRET (General Purpose REGister RETAined) is a RAM mapped retained register which preserves its value even after the system reset. In order to make the bootloader enter the DFU mode, a special reserved value must be written to this register.
A connected client can turn on the DFU mode by writing to a characteristic of a vendor-specific BLE service called *Secure DFU Service* (see section 5.4.4).

The service then writes to the GPREGRET register and executes a system reset [15].

After the device resets, the DFU mode is entered, the bootloader starts advertising with the name *SunFibre - DFU*, and waits for an init packet. No authentication (bonding) procedure is required to connect and start the firmware upgrade. An inactivity timer is turned on, and if it expires, the bootloader resets back to the application.

5.10.4 Firmware Upgrade Process

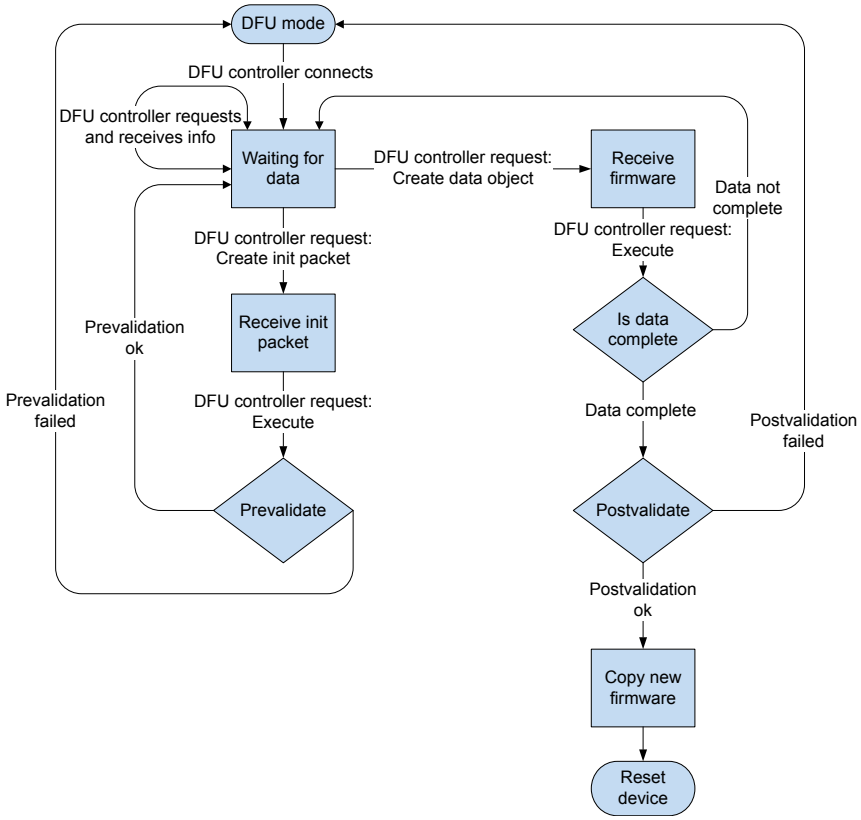


Figure 5.12: WorkFlow of the DFU process (taken from [16])

Two devices are involved in the DFU process - the DFU controller, which transfers the DFU package (smartphone), and the DFU target (light control unit or floating button), which receives and applies the new firmware [16]. The whole DFU process is visualized in figure 5.12.

Init Packet

At the start of the DFU process, the DFU controller transfers the init packet. It contains various fields describing the content of the DFU package (such as the type of package - in our case, the application only, hardware/firmware version, SoftDevice ID etc.). A signature generated using a private key is also added to the packet to ensure the authenticity of the firmware provider. Additionally, a hash of the whole new firmware created by the SHA256

function is appended.

For signing the DFU upgrades, a completely new private key was generated by open-SSH.

Prevalidation
Upon successful reception of the init packet, the prevalidation is performed. At first, the signature of the DFU package is verified with the public key compiled directly in the bootloader code. Subsequently, the hardware version is checked as well as the firmware version, which prevents a possible downgrade.

Data Transfer
The DFU process is configured as a dual-bank update. The existing application is preserved until the new firmware image is activated. If the firmware update process fails, the user can still restart the device and boot the existing application.
The data transfer is carried out according to a documented Nordic protocol using a BLE service on the DFU target called *DFU Service*. It is a primary service with a 16-bit UUID - **0xFE59** recognized by Bluetooth SIG. The DFU controller initiates the DFU operations (init packet transfer, prevalidation, data packet transfers etc.) by writing commands to *DFU Control Point Characteristic*. A response from the DFU target comes as a BLE notification. The package is split into multiple chunks (packets) that are consecutively written to the the *DFU Packet Characteristic* and transferred [17].
The upgrade status is preserved in a dedicated flash memory region - MBR Params allowing the upgrade process to continue even when it was suspended (5.2).

Postvalidation
Upon successful reception of the whole DFU package, the bootloader checks the hash of the received application image to verify the integrity of the data. If it passes, the new firmware is activated - copied from the second bank to the first bank, and the application is booted up.

5.10.5 Tools Used

Nordic offers a free application - *nRFToolBox - DFU* that implements the transport protocol and helps the developers to test the bootloader and DFU. Eventually, the DFU functionality was added to the custom mobile application too - see section 7.3.3.

The init packet or bootloader settings generation is possible with the help of *nrfutil* [20], a python based command-line tool also provided by Nordic.

Chapter 6

RFID Tag Detection

This chapter comments the integration of RFID readers introduced in section 4.3 and explains the software techniques and algorithms for RFID tag detection. A general description of RFID tags is provided in section 3.2.1, the introduction to the EM4100 RFID tag in section 4.3.3.

6.1 EM4100 Protocol Description

The EM4100 memory array is consisted of 64 bits of five different types (see figure 6.1). The first 9 bits are used as synchronization header and all are preprogrammed to logic 1. The header is followed by 10 groups of 4 data bits (D00-D93) and 1 even row parity bit (P0-P9). Each group represents one number of tag id. The last group consists 4 even column parity bits (PC0-PC3) and 1 stop bit set to logic 0 [4].

1	1	1	1	1	1	1	1	1	9 header bits
8 version bits or customer ID				D00	D01	D02	D03	P0	
				D10	D11	D12	D13	P1	
				D20	D21	D22	D23	P2	
32 data bits				D30	D31	D32	D33	P3	
				D40	D41	D42	D43	P4	
				D50	D51	D52	D53	P5	
				D60	D61	D62	D63	P6	
				D70	D71	D72	D73	P7	
				D80	D81	D82	D83	P8	
				D90	D91	D92	D93	P9	10 line parity
				PC0	PC1	PC2	PC3	S0	bits

Figure 6.1: EM4100 memory array (taken from [4])

Data Modulation:

As discussed in the section about the principles of RFID (3.2.4), the tag transmits the data by modulating the RF field of the reader. The data transmission is synchronized by individual cycles of the RF field used (125 kHz). The length of every transmitted bit can be either 32 or 64 cycles. This makes the data rate approximately 4 kHz in the case of 32 cycles per bit.

Data Encoding:

The EM4100 data are further encoded by Manchester encoding producing a logic level change in the middle of the bit period. A low to high transition represents a logic 1, while a high to low transition represents a logic 0 [4].

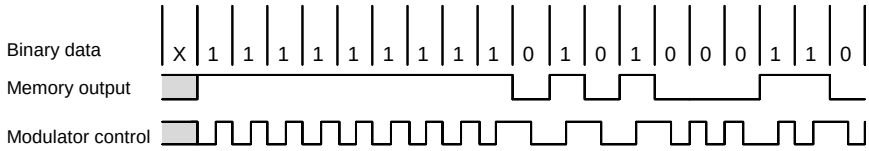


Figure 6.2: Manchester encoding (taken from [4])

6.2 RDM6300 RFID Reader Data Processing

As described in section 4.3.1, the RDM6300 performs all signal processing (demodulation and decoding) of the EM4100 protocol internally. Once the RFID tag is detected, it only transmits the processed tag id with some synchronization bytes and checksum over UART. Because the data is sent only once, the tag must be moved out from the reader's detection range to be transmitted again.

The 64 bits saved in the RFID tag are sent as a 14 bytes long word (see figure 6.3). The head (first) byte of the word is always set to **0x2**, and the tail (last) byte to **0x3**. After the head, ten bytes representing the tag's unique identifier and two bytes of checksum follow. They are sent as hexadecimal characters in ASCII format.

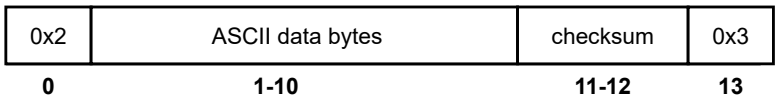


Figure 6.3: RDM6300 word

The checksum is calculated as XOR operation against each pair of the data bytes converted to integer values. If the calculated checksum matches the transmitted checksum, the RFID word is valid.

6.3 LC RFID Reader Control and Signal Processing

The introduction to the custom LC RFID reader with the circuit scheme is in section 4.3.2. To remind the principle, this RFID reader is based on synchronization between excitation of LC oscillator by PWM (on resonance frequency 125 kHz) and sampling of the RFID carrier signal. The signal is synchronously sampled at the carrier amplitudes on the sampling capacitor (C_3 in scheme 4.13) to reach the highest sensitivity, and then it is processed to get the tag id.

Three PPI channels were used to off-load MCU and to minimize the latencies between timer compare events and triggering tasks - GPIO set and ADC sample task (see 4.1.1). The first PPI channel sets the PWM pin high, together with resetting the timer, and it is triggered by compare event generated when the timer reaches a value corresponding to the period of the RFID signal. The second channel sets the PWM pin low in the middle of the period and the third channel starts ADC sampling with some latency (offset) to the start of the PWM.

6.3.1 Sampling

The phase shift between PWM generator and sampling capacitor (C_3 in scheme 4.13) might be calculated but it was found experimentally.

The frequency of nRF52833 timer is $f_t = 16\,\text{MHz}$. It is reset when its value reaches the period of RFID signal - 128 ticks (f_t/f_{RFID}). This allows setting programatically the latency of the sampling with resolution $\Delta t = \frac{1}{f_t} = 62.5\,\text{ns}$. To get the optimal latency, an average of samples over multiple periods for every possible timer value (1-128) was calculated, and the maximum was found.

The ADC configuration is shown in table 6.1 below. The acquisition time must be lower than the period of sampling. The gain was set with respect to the amplitude of voltage on the sampling capacitor when no tag was present.

Sampling Frequency	125 kHz
Resolution	10 bit
Acquisition Time t_{acq}	5 µs
Reference	internal 0.6 V
Gain	1/2

Table 6.1: ADC channel parameters for RFID sampling.

6.3.2 Signal Processing

The signal processing comprises demodulation and decoding of the raw ADC samples and finally constructing the RFID tag id from the decoded EM4100 data (see figure 6.5a showing all steps).

Demodulation:

The goal of demodulation lies in the extraction of Manchester encoded data (refereed further as half-bits) from the raw samples (symbols). The algorithm of data demodulation is a single-pass algorithm that loops over the samples and outputs a buffer with the half-bits.

To determine the logic level of a particular sample, the moving average crossover technique was used - a comparison of two moving averages, one with a short and the second one with a long window. The short window smooths the signal, while the latter calculates a comparison level. Generally, the comparison level might differ significantly depending on the tag type

(how much energy of the RF field it draws), distance from the reader or surrounding environment.
After every logic level transition, either one or two half-bits are appended to the output buffer because the Manchester encoding allows only two consecutive half-bits of the same logic level, and the length of each half-bit is fixed - 32 symbols (carrier cycles).

The lengths of windows were optimized during the development and, finally, the lengths $K = 256$ for long window and $L = 16$ for short window were used.

Decoding:

In this phase, the demodulated half-bits are processed in pairs and the data-bits are decoded. Sometimes a forbidden combination of logic levels appears (logic 11 or 00). It could mean that the first half-bit is odd and must be discarded or if it appears for the second time that the data are corrupted.

Finding Tag ID:

At first, the processing algorithm looks for a synchronization header, a group of nine logic 1. Subsequently, it converts the data bits to tag id while controlling the row and column parities. If the header is not found or any of the row or column parity bits are wrong, the data are discarded.

6.3.3 Detection

A satisfying technical realization of RFID detection offers low latency (response time) and maintains low battery consumption. However, these properties go against each other.

Battery Consumption:

The battery consumption is influenced mainly by generating PWM to excite the LC oscillator and should be kept as low as possible. For this reason, 320 test samples are regularly taken to check the potential presence of an RFID tag. If there is a significant voltage difference between the test samples which could indicate logic level changes, the ADC starts sampling for a sufficiently long period to cover one whole EM4100 word (64 data bits). The sampling is performed without MCU involvement, and once it is done, the signal is processed as described in the subsections above.

Response Time:

To achieve a reasonable response time, the presence of a tag is checked every 150 ms. Meanwhile, all peripherals for RFID detection are turned off – timer, PPI, EasyDMA and ADC.

Number of Samples

One EM4100 word has 64 data-bits, Manchester encoding doubles their number, and the symbol rate of EM4100 modulation is 32 carrier cycles per bit. As a result, a minimum of 4096 ADC samples is needed. However, this number was increased for two reasons. First, the samples at the beginning when the tag gets charged usually differ a lot in amplitude, so it is better to remove them for more accurate detection. Second, for the further signal

processing and the header finding, it is beneficial to sample at least one group of nine synchronization ones together, consecutively.
In the end, this increased the sampled number of bits to 5096.

6.4 RFID Application Logic

Regardless of using RDM6300 or LC RFID Reader, the application logic is the same. The RFID detection is turned on, only if enabled through the BLE characteristic *RFID Enabled* (see 5.4.4). Disabling RFID detection significantly helps to improve the battery life (see the measured battery consumption in section 9.1.2).

The application event EVT_RFID_DETECTED is generated upon receiving a valid RFID tag id, the BLE characteristic *RFID Detected Tag ID* is updated, and all connected clients are notified if they enabled notifications.
The light mode is incremented only if the detected tag id matches the content of the BLE characteristic *RFID Paired Tag ID* (see methods for changing light modes in section 2.3.1).

Both characteristic values, *RFID Enabled* and *RFID Paired Tag ID* are preserved in the flash memory in a section before the bootloader. The backup is done to avoid losing the configured properties during device resets.

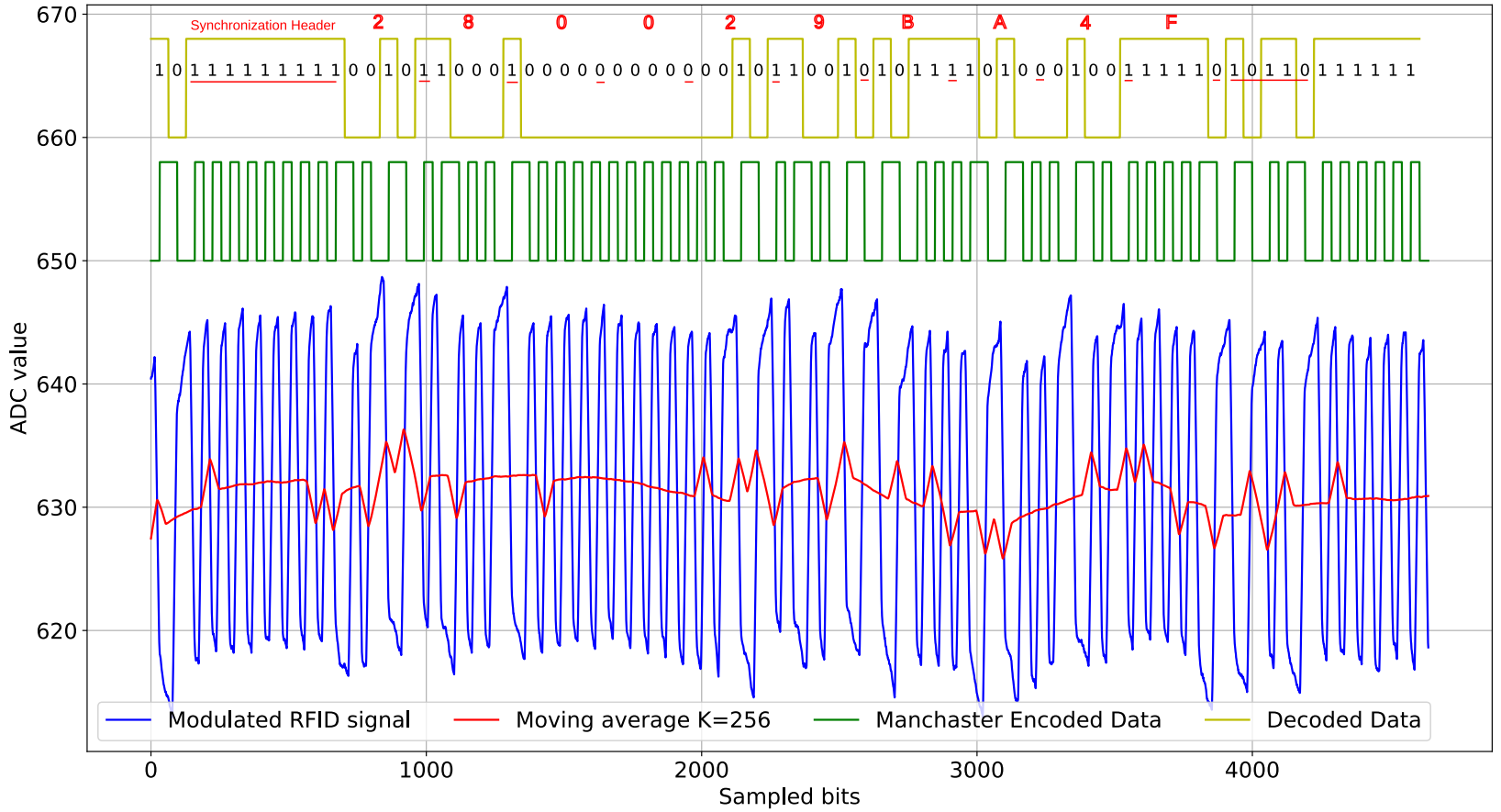


Figure 6.4: EM4100 RFID signal processing

(a) : Bottom part: sampled amplitudes of LF RFID signal of EM4100 at 125 kHz (blue line) with comparison level (red line) based on a moving average window of 256 samples. Upper part: Manchester encoded (green line) and decoded (yellow line) data bits. The control bits - synchronization header, parity bits and the stop bit are underlined in red. The resulting serial number of RFID tag is **280029BA4F**.

Chapter 7

Software Development - Mobile Application

This chapter summarizes all necessary information about the custom mobile application. Apart from that, it introduces different approaches to mobile application development and compares them.

7.1 Mobile App Development

There are multiple approaches to mobile application development - native, web and hybrid. Because each one has some downsides and upsides, this section provides a small comparison. Due to the growing popularity of React Native and previous experience with web development techniques, the hybrid approach was chosen.

7.1.1 Native Applications

Native mobile applications are specifically developed and built for one platform. They use platform-specific programming languages, software development kits, and development environments offered by the OS providers [37]. For example, Apple's iOS uses Swift programming language, whereas Android uses Java or Kotlin.

Pros:

- Performance and reliability
- All OS features support
- Full access to the HW features

Cons:

- A separate development for each platform
- Complicated maintenance

7.1.2 Web Applications

In a nutshell, mobile web applications are basic web applications (websites) but adjusted for small screens. They don't need to be installed on the device, but they are rendered in a device's web browser. Web technologies such as HTML5, CSS and JavaScript are used for their development.

Pros:

- One development for all platforms (including desktop)
- Web technologies (Javascript)

Cons:

- No access to the HW features
- Limited support of OS features
- The need for a web browser

7.1.3 Hybrid Applications

Hybrid mobile applications combine the strengths of the two previous approaches. To build a hybrid app, developers write the core of the application with the help of web technologies but place a native device wrapper around it [37]. This is typically ensured by a framework which provides a set of JavaScript APIs to access device features through native APIs.

Pros:

- One development for all platforms
- Fast development

Cons:

- Limited access to the OS and HW features
- Performance and reliability

7.2 Tools Used

The mobile application was written in JavaScript wrapped by React Native framework. All BLE functionality is handled by a third-party react-native-ble-plx library. Expo toolchain was used to simplify the development operations - building and deploying the application releases on Android and especially iOS. A special platform called TestFlight for beta application testing on iOS was also needed.

7.2.1 React Native

React Native is a JavaScript framework for developing mobile applications that can run natively on Android and iOS. It is based on ReactJS, created by Meta Platforms, a declarative, component-based framework for developing web user interfaces. The code, written in JavaScript, is platform-independent, reacting with UI components or libraries that get compiled into the native code. Additionally, attaching some platform-specific (native) code is always possible. However, the JavaScript code is not being compiled to the platform's native language and is run by the JavaScriptCore framework that provides the direct access to WebKit's JavaScript engine [28].
The user interface of an application is assembled from React components, JavaScript classes or objects. In case of application state change, only those React components that contain modified data re-render.

react-native-ble-plx

React-native-ble-plx [29] is a React Native library that allows using the

iOS/Android Bluetooth Low Energy APIs and thus communicating and controlling the BLE peripherals.

7.2.2 Expo

Expo is a framework and a platform for React Native apps. It is a set of tools and services built around React Native and native platforms that helps the developer to build, deploy and test mobile applications.

7.2.3 TestFlight

TestFlight is an online service and iOS/macOS application for installing and testing the beta versions of mobile applications offered to developers within a paid iOS Developer Program.
At first, the tester must download the TestFlight application from AppStore. The developer then invites the testers based on their Apple ID and provides them with particular builds (releases) for testing.

The testing and installation of the application on Android is much easier. The tester can install the application from the .apk file provided by the developer directly.

7.3 UI Structure

The mobile application contains three top-level screens - the introduction screen, devices screen and device control screen. The component diagram showing the relationships between components is visualized in figure 7.7. The devices and device control screen are not accessible unless all required app permissions are enabled and Bluetooth is turned on. The application supports communication with multiple light control units, but only one can be controlled.

The functional requirements are specified together with use cases in following subsections that correspond to the top level screens.

7.3.1 Introduction Screen

The introduction screen is shown when the application loads. It exhibits the SunFibre brand and provides some information about the product. Apart from that, all necessary application permissions are checked, and it prompts the user to turn on Bluetooth if not enabled previously.

7.3.2 Devices Screen

Devices screen displays a list of all available devices (both advertising devices and already connected devices). It allows the user to connect to a new device or open a control screen of an already connected one.

Every device item in the list contains a device name or saved local name (check the paragraph about local device name in the next section), MAC

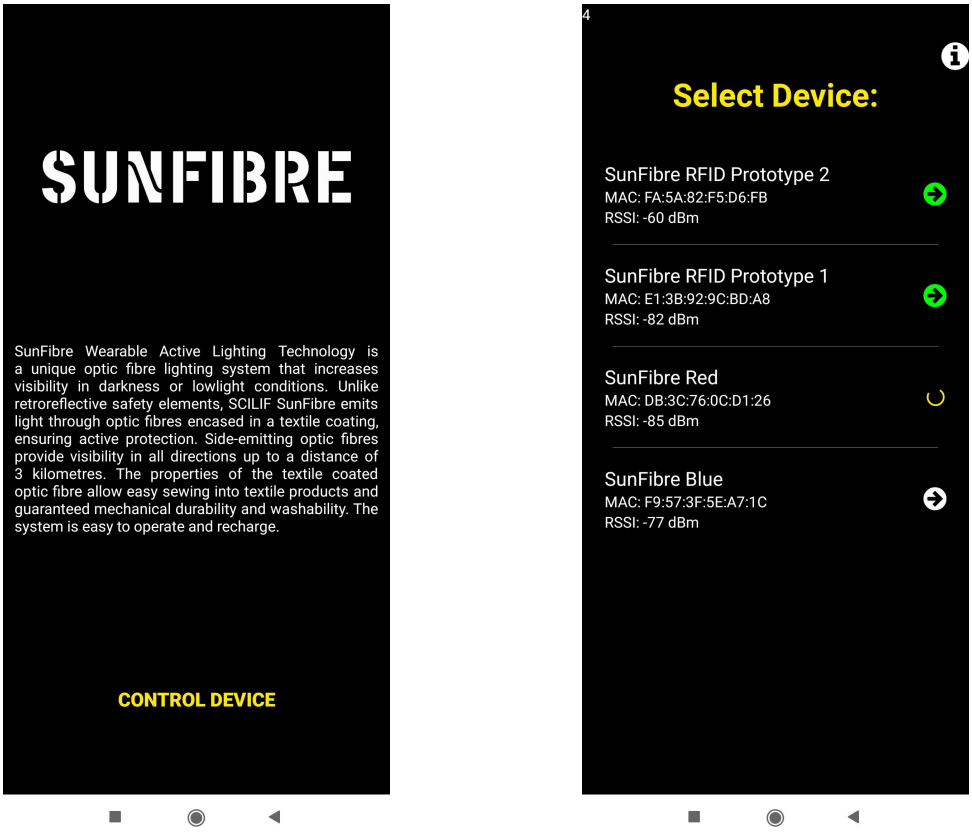


Figure 7.1: Introduction and devices screen

address, RSSI and navigation icon. The icon color determines the connection status, and when it is clicked, the user is redirected to the control screen.

The list of devices can be refreshed by a swipe down.

7.3.3 Device Control Screen

Device control screen provides all information about the light control unit and allows the remote control and OTA DFU upgrade. By default, only the unit name, battery level, charging status and current light mode are displayed, but additional monitored data are available in dialogs.

Light Mode

The user can change the current light mode on the light control unit by pressing one of the control buttons (the current mode is highlighted). If the light mode gets changed on the unit, it is also automatically changed in the application.

Local Device Name

Because all devices have the same predefined name which they advertise, the application allows using an arbitrary local name to distinguish them better. The local name is preserved in the application memory and is used in future.

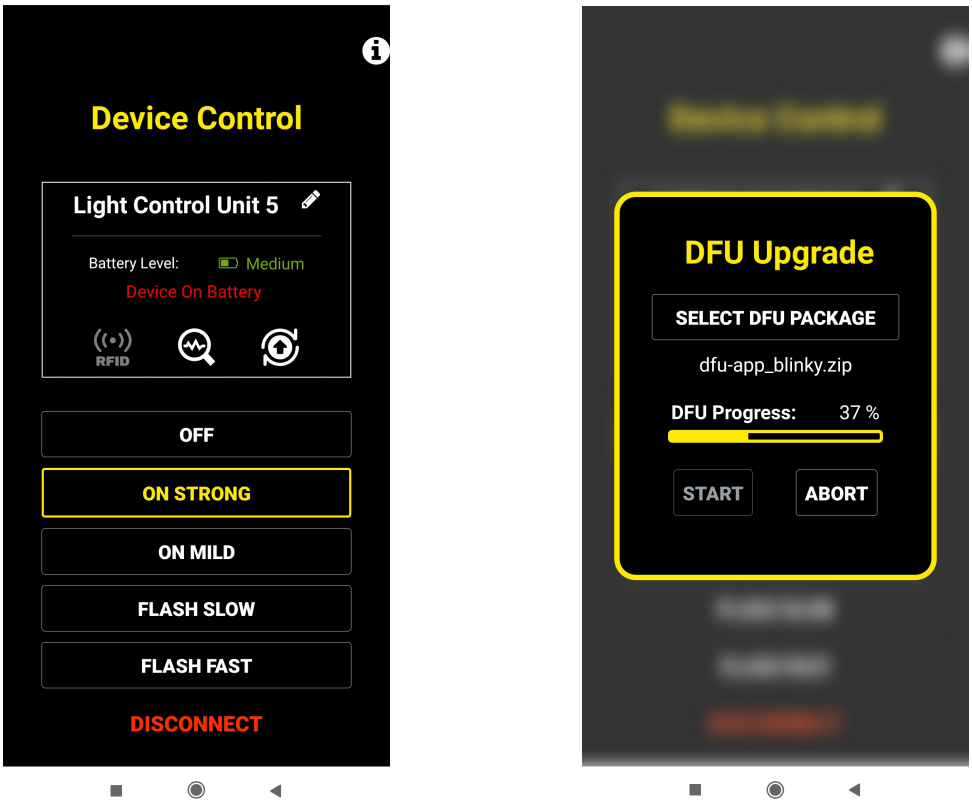


Figure 7.3: Device control screen and DFU upgrade dialog

Disconnection

Click on the last control button disconnects the selected light control unit. If the user doesn't terminate the connection but the unit disconnects, the application automatically closes the device control screen and redirects the user to the devices screen.

Monitor Dialog

Monitor dialog shows additional monitored data and device information, such as LED voltage and current, battery voltage, MCU temperature and firmware and hardware version. It can be activated by clicking the monitor icon.

RFID Monitor Dialog

As long as the device has the RFID BLE service, the RFID monitor dialog can be opened by clicking on the RFID icon.
The dialog allows the user to configure the RFID functionality - turn on/off the detection and change the paired tag ID for light mode changing. Regardless of paired tag ID, it shows which tag was detected the last time and when.

DFU Upgrade Dialog

The DFU upgrade dialog can be utilized for over-the-air firmware upgrades. Initially, the user must select the DFU package from the phone's storage. Afterwards, the OTA upgrade starts by activating the DFU mode on the

connected device. The device resets to the bootloader, and a scan is performed in the background to find and connect the advertiser with the name *Sunfibre - DFU* and the same MAC address. Subsequently, the application transfers the init packet and the actual application firmware in multiple chunks via the Nordic transport protocol. The DFU upgrade process is described in detail in section 5.10.4.
When the firmware upgrade completes, the user is redirected back to the devices screen

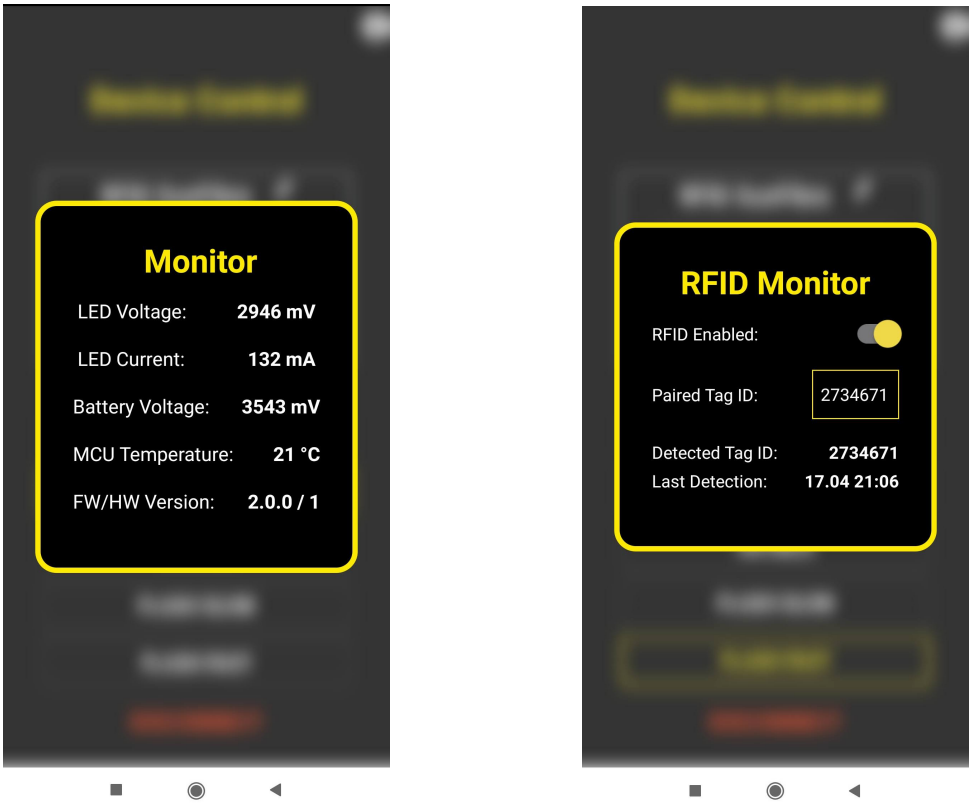


Figure 7.5: Monitoring dialogs

7.4 BLE Implementation Details

All BLE methods written in verbatim below are asynchronous, provided by the react-native-ble-plx library. The library returns JavaScript Promises.

The devices screen uses the `start/stopDeviceScan` BLE function to start scanning. The scan is started when the screen is opened and closed when the user navigates elsewhere.
Because the connection and bonding procedure is handled internally by the native APIs, when a new device is connected via `connectToDevice` method, the programmer can not determine whether bonding is required or not.
After a successful connection, a device services and characteristics discovery is performed through `discoverAllServicesAndCharacteristicsForDevice`

function.

The control screen retrieves data from BLE characteristics defined in section 5.4.4. Some characteristics are read via `readCharacteristicForDevice` only once - FW/HW version. Some are periodically polled - temperature, LED current, battery voltage etc.
A few characteristics are monitored via `monitorCharacteristicForDevice` and asynchronously updated when a notification is received - light mode or detected RFID tag ID.
In the case of user input, the corresponding BLE characteristics are written immediately by `writeCharacteristicWithoutResponseForDevice` - for example, light mode, paired tag ID, or RFID enabled.

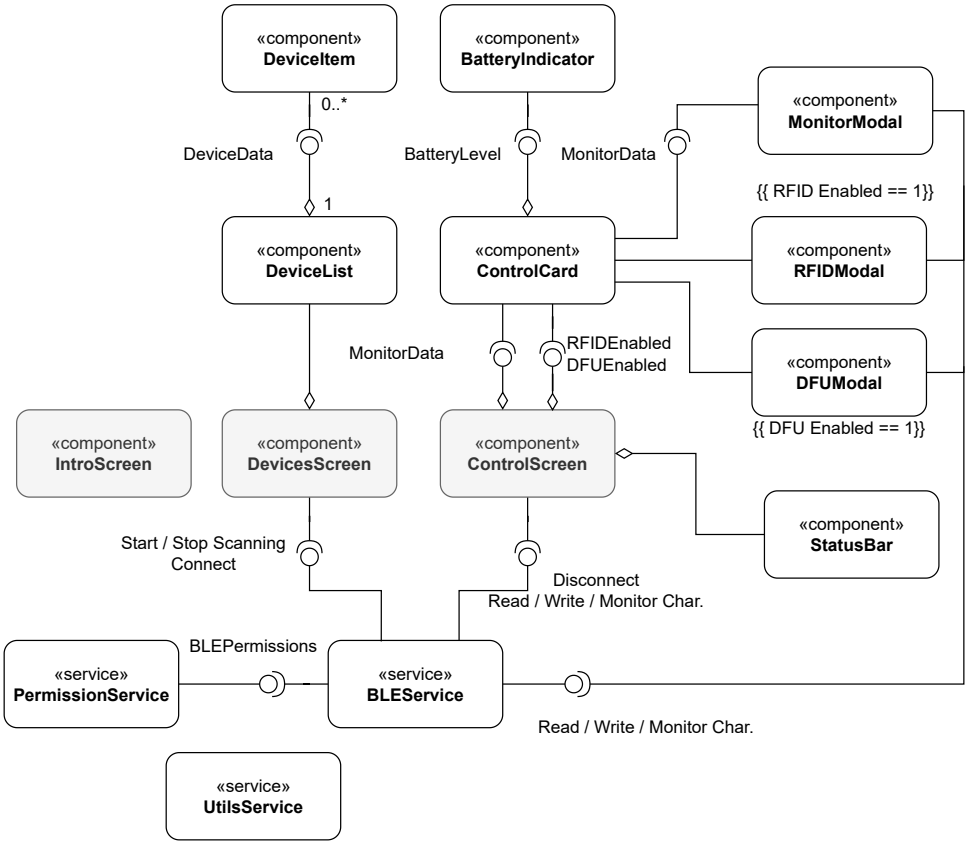


Figure 7.7: UML component diagram of the mobile application, created according to [39]. The aim was to completely split the BLE functionality from the UI logic into a separate service. The top-level screen components are routed from the main App.js file.

7.4.1 Limitations

Although react-native-ble-plx offers a comfortable and fast way how to work with BLE peripherals, it lacks some crucial functionality that makes a further transition to native APIs inevitable. Namely:

- No way how to get system (OS) connected BLE devices
- No support for scanning and connection parameters setup
- No control over bonding
- No support for smart watch devices.

Especially the last problem does not have any straightforward solution. There is currently no support for React Native with BLE functionality for smartwatch applications. Because an iWatch application is one of the future wishes of the client, the application is going to be rewritten in native programming languages in near future.

Chapter 8

Gesture Control

The achieved progress in gesture control, as one of the light mode change methods (2.3.1), is briefly commented in this chapter. Due to the lack of time and the overall complexity of the task which might be a topic for another thesis, the work on gesture control only started. The hardware functionality was tested, the data-set collected and the first scripts with machine-learning classifications of recorded sequences written.

Considering the possible use cases, the gesture control could be either implemented on the light control unit with a wired external sensor, or used directly in the floating button. However, this depends on the floating button position.

8.1 Inertial Gesture Recognition

Human hand gestures are a widely accepted form of real-time input for human-machine interfaces. Most of the previous work on gesture recognition has been based on computer vision techniques, however, the inertial sensors (accelerometer & gyroscope) are used in some papers too.

The main machine-learning algorithms employed in the literature for inertial gesture recognition are DTW and HMM. Both process the data in the time domain. The **Dynamic Time Warping (DTW)** is a procedure to find similarities between two sequences of values. A **Hidden Markov Model (HMM)** is a statistical Markov Model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. While having a reasonable accuracy, these approaches have also serious disadvantages regarding the implementation on a MCU. Data processing and feature extraction techniques might be a favourable approach for MCU implementation [38], [6].

8.2 Data-set Collecting

Finally, the light control unit with the accelerometer/gyroscope LSM6DSMTR (4.1.6) was exploited for data collecting instead of the floating button which has only an accelerometer.

A test firmware that retrieves the data from the unit was written. The user presses the button, grabs the unit to the hand, and waits for a LED signal to perform a specific gesture for 10 seconds. The collected data are sent via UART to the computer. Later, the data are preprocessed, converted from raw values and saved locally for later processing.

Data for five different gestures were collected - wrist circles, arm circles, front arm rises, overhead arm rises and arm extending left (cyclist left turn).

Output Data Rate:	12.5 Hz
Acc./Gyro Full Scale:	$\pm 2g$ / $\pm$ 250 dps
Acc./Gyro Sensitivity:	0.061 mg/LSB / 7.63 mdps/LSB
Acc./Gyro Bandwidth:	400 Hz / 351 Hz

Table 8.1: LSM6DSMTR parameters for gesture control

8.2.1 LSM6DSMTR Setup

The sensor communicates with the nRF52 over the I^2C bus interface as a slave. Because the gestures are rather slow and long events, the data rate, sensitivity and bandwidth were adjusted accordingly - table 8.1.

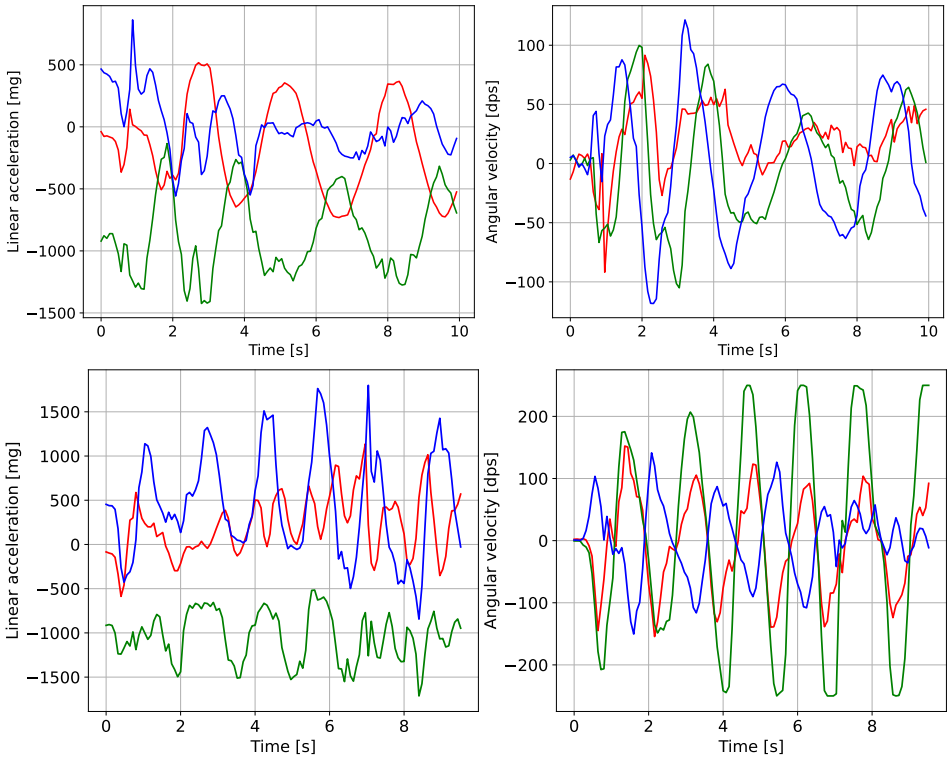


Figure 8.1: Examples of recorded gestures: clock-wise wrist circles & overhead arm rises

Chapter 9

Results

The objective of this chapter is to validate the implementation and requirements, show achieved results and discuss performed tests.

9.1 Battery Consumption

To validate the requirements for battery life and selection of parameters, such as connection and advertising intervals, a detailed analysis of battery consumption for the light control unit was performed.

We can divide the sources that drain the battery into static and dynamic. The static sources have a constant effect on battery consumption, whereas the dynamic sources are repeated pulse events occurring within a certain period.

The static consumption was measured by a digital multimeter **Tenma 72-7730A** connected as an ammeter between the battery jumper - the positive battery pin and the stabilizer. The multimeter has better resolution when measuring current (0.01 µA) than voltage.
The electrical charge drawn by one occurrence of a dynamic event was sensed on a 10-ohm shunt also connected between the battery and stabilizer by a 1 GSa/s, 8-bit **Rigol MSO1104Z** oscilloscope. The raw oscilloscope values were saved as a binary .wfm file and later retrieved by a python library RigolWFM [31], preprocessed and numerically integrated.

The measured values, especially the estimated durations, should provide only a rough estimate but not precise values.

9.1.1 Static Consumption

The major sources of the steady consumption are the nRF52833 chip, powered board parts and PCB leakage currents.

In order to verify the proclaimed current consumption of the nRF power modes, a simple test program that enters a particular mode without peripherals initialization was written (first three rows in the table 9.1.1). The power management of nRF52 is briefly described in section 4.1.1.

Mode	Avg. Current
System-OFF	7-9 µA
System-ON	7-9 µA
Max. Performance	5.67 mA
Deep Sleep	7-9 µA
IDLE	16-19 µA

Table 9.1: Average current consumption of nRF and application states

Peripheral On	Avg. Current
Red LED	663 µA
Green LED	440 µA
Blue LED	350 µA
RDM6300	15.4 mA
Accelerometer	x
Gyro	x

Table 9.2: Average current consumption of board peripherals

The minimum measured current consumption is between 7 and 9 µA regardless of the nRF power mode. Consequently, we can deduce that this consumption is mainly made up of the powered board peripherals and PCB leakage currents. The consumption listed below makes roughly the measured value.

- nRF52833 - 1.5 µA (System-ON), 0.6 µA (System-OFF) [24, p. 56]
- LSM6DSMTR accelerometer/gyroscope - 3 µA [9, p. 26]
- MCP1700T stabilizer - 1.6 µA [10, p. 1]
- DIO5661ST6 - 1 µA [3, p. 1]
- Battery measurement leakage current <2.86 µA (see 4.1.3)

The last two measured consumption in table 9.1.1 refer to the application states SYSTEM OFF (Deep Sleep) and IDLE. In IDLE, the power consumption is increased because all necessary peripherals are initialized (BLE, PWM, ADC, RTC, button, GPIOs etc.), but the nRF is sleeping, waiting for an event (see section 5.6.3 showing the application states).

9.1.2 Consumption of Periodic Events

To express the battery consumption of a periodic event, both technically and demonstrably, three values are presented. The first one Q_{EVT} denotes how much battery charge in microcolombs (µC or µAs) gets consumed by one event occurrence, and the second one I_P stands for the peak current. The last one determines how quickly the event consumes 10 % of the battery, considering a fixed period and Deep Sleep mode in between.

The drawn electrical charge from a battery can be described by the equation below:

$$Q(t,T) = t\,\frac{Q_{EVT} + (T - D)\,I_{OFF}}{T}, \tag{9.1}$$

where D denotes the duration of the event and I_{OFF} is the current consumption in Deep Sleep mode.

When we assume a fixed period and $Q(t)$ equal to 10% of a typical lithium battery - 160 mAh, we get the expected duration. It is further denoted by $t_{10\%}$ and the units are days.

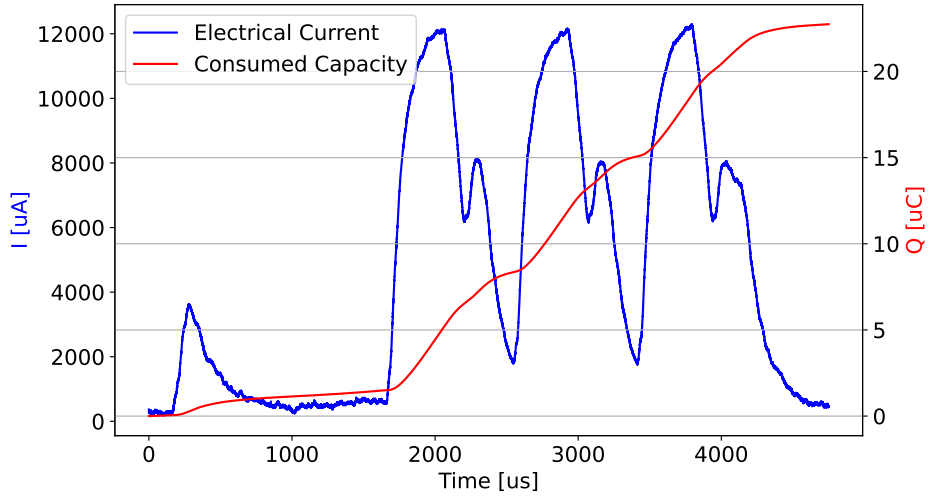
BLE - Advertising & Connection

In this experiment, a comparison of consumed battery capacity for advertising and connection packets is shown. The periods (advertising and connection interval) in table 9.3 are those defined in application firmware (check table 5.2).

Considering our use case, there is no data to send most of the time during a connection. Therefore, the empty data packet was measured, and thus the event takes less energy. The TX power level was set for both to 0 dBm.

Event	Q_{EVT} [µC]	I_P [mA]	$t_{10\%}$ [d]
Fast Advertising $T = 100\,\text{ms}$	22.74	12.30	28.36
Slow Advertising $T = 1000\,\text{ms}$			217.11
Initial Connection $T = 100\,\text{ms}$	9.28	10.86	66.27
Connection $T = 250\,\text{ms}$			148.01

Table 9.3: Battery consumption - advertising & connection



(a) : First peek is the start of MCU and HFCLK oscillator, receive (RX) and transmit (TX) amplitudes on three channels follow.

Figure 9.1: Battery consumption - advertising packet

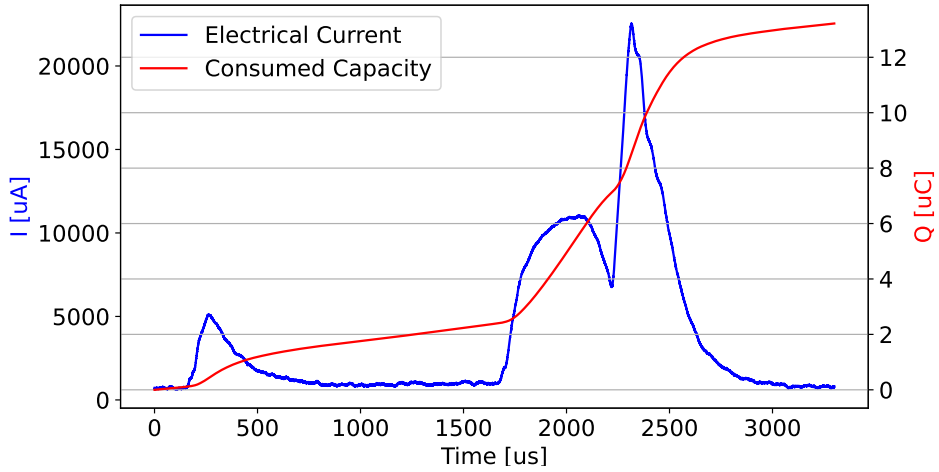
BLE - Connection TX Power Levels

This experiment compares the battery consumption of radio with a connected antenna for a different TX power level setting during a connection. The RX power level is always the same and cannot be changed via SoftDevice (Nordic BLE stack).

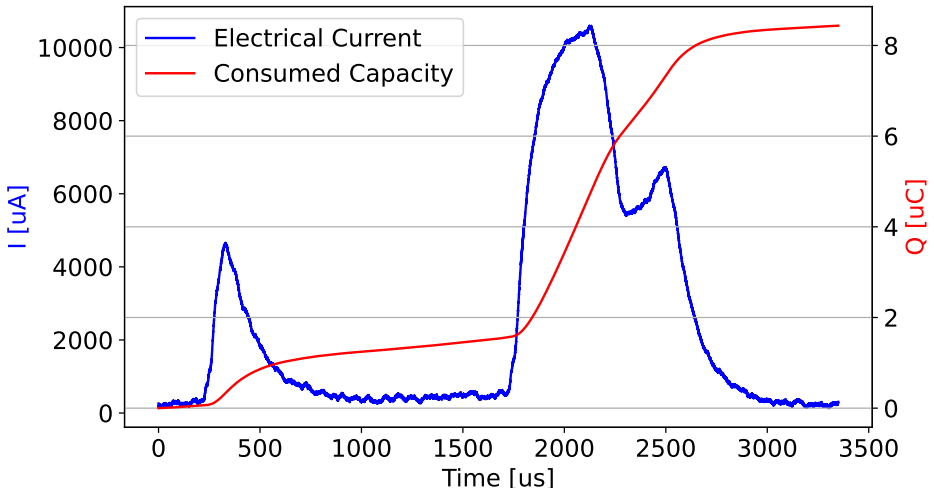
A constant part of the total energy is needed for activation of the oscillator, radio as well as for packet receiving. For TX power level of 0 dBm and lower, the energy needed for transmission becomes less significant and the RX amplitude dominates.

Connection	Q_{EVT} [µC]	I_P [mA]	$t_{10\%}$ [d] $T = 250\,\text{ms}$
+8 dBm	13.21	22.54 (TX)	109.70
+4 dBm	11.63	16.06 (TX)	122.51
0 dBm	9.28	12.30 (RX)	148.01
-4 dBm	8.78	10.60 (RX)	154.85
-8 dBm	8.43	10.55 (RX)	160.11
-16 dBm	8.36	10.41 (RX)	161.35

Table 9.4: Battery consumption - different connection TX power levels



(a) : +8 dBm: The first peek is the start of HFCLK oscillator and MCU. The communication runs on one channel but only empty packet is sent. The transmit (TX) amplitude is dominant.



(b) : -16 dBm: Small transmit (TX), dominant receive (RX) amplitude.

Figure 9.3: Battery consumption - +8 dBm, -16 dBm data packets

ADC - RFID Check and Detection & Monitor

This test shows the measured consumption of events when ADC is involved - monitoring (battery, LED and temperature measurement) and RFID check and detection. During the check of the RFID tag presence, the tag was not near the coil. Therefore, no energy was taken via inductive coupling (see section 3.2.1 about the principle of RFID tags). The RFID detection (shown in figure 9.4) consists of the RFID check, sampling and decoding the tag word. In this case, the tag was a few centimeters from the coil.

The periods used for RFID check and monitoring are the same as in current firmware (see sections 6.3.1 and 5.9.3). Even though the RFID detection is not a periodic event, the lowest possible period was used because the detection gets disabled for one second if a tag is detected.

Event	Q_{EVT} [µC]	I_P [mA]	$t_{10\%}$ [d]
RFID check $T = 150\,\text{ms}$	47.35	17.46	20.60
RFID detection $T = 1000\,\text{ms}$	737.79	16.49	8.94
Monitoring measurement	2.87	5.66	744.52

Table 9.5: Battery consumption - RFID & monitoring

9.1.3 Consumption in Application States

In this subsection, a breakdown of battery consumption in application states - JOINING ADVERTISING and CONNECTED is shown. The behaviour of the light control unit in these states is described in section 5.6.3.

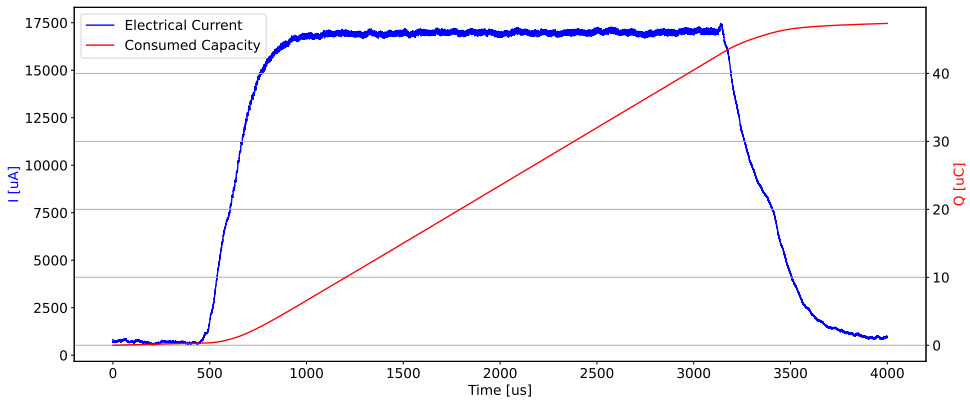
The total consumed battery capacity Q_{TOT} is calculated for 60 seconds of each state. For the sake of simplicity, we assume that the battery is fully charged - green color of *battery LED*, no tag is close to the unit - no RFID detections, the TX power level is for both to 0 dBm and no responses are coming either from a scanner if advertising or a master when connected. Further, we assume that the nRF is sleeping (System-ON) between events.

The results - Q_{TOT} and perceptual distribution are provided in table 9.6.

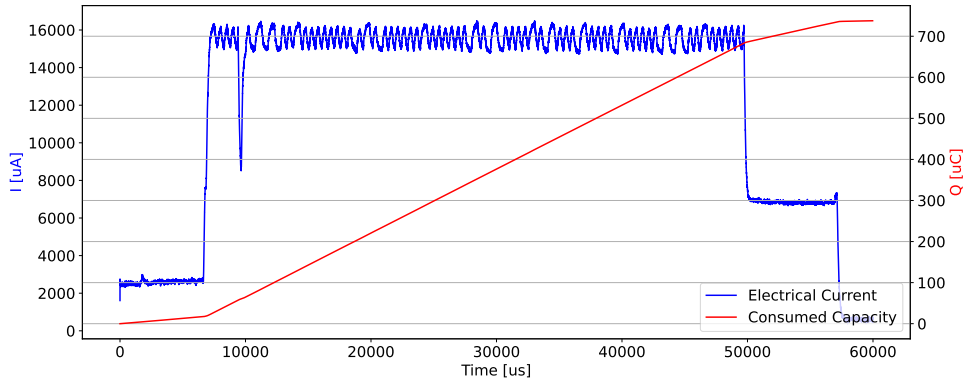
9.1.4 Conclusions

This subsection analyzes the conclusions from previous tests.

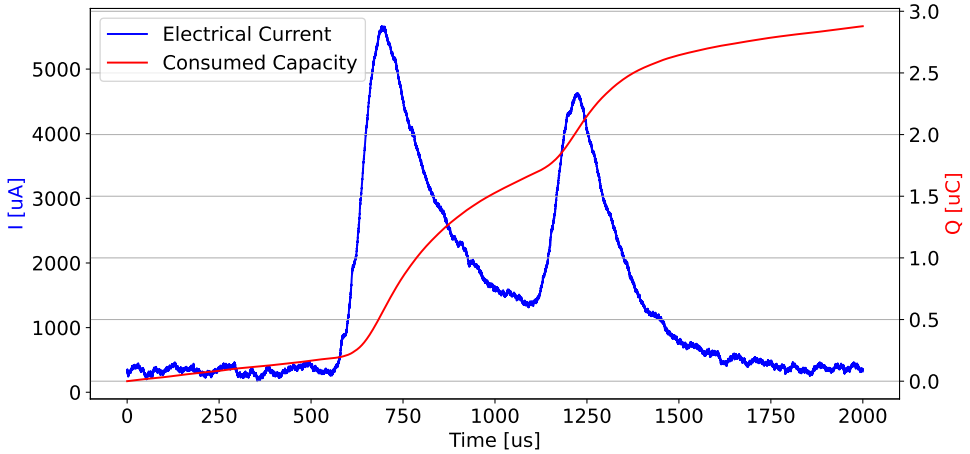
It is evident that the measured and discussed values and possible optimization play an important only if the lighting is turned off. If any of the lighting modes defined in section 2.3 is active, 10 % of the battery is drained in less than 3 hours. From this perspective, the remote control brings negligible overhead.



(a) : RFID check: 320 samples at 125 kHz, the inflexion point on the falling edge is a result of short data processing .



(b) : RFID detection: first 2.56 ms long part till the sharp fall is the RFID check (same as figure (a)), second 40.7 ms part is sampling of 5096 samples at 125 kHz and the third 7 ms part is data processing - demodulation and decoding. See section 6.3.1.



(c) : Monitoring: the first maxim is the start of ADC and HFCLK, and the second maxim is the actual sampling of 5 samples of 3 ADC channels. See section 5.9.3.

Figure 9.4: Battery consumption - RFID & monitoring

	JOINING ADVERTISING		CONNECTED	
	Q_{TOT} [µC]	Q_{TOT} [%]	Q_{TOT} [µC]	Q_{TOT} [%]
System-ON	1016	1.27	1018	1.38
RFID Check $T = 250\,\text{ms}$	18940	23.72	18940	25.59
BLE Adv. $T = 1000\,\text{ms}$	1364	1.71	x	x
BLE Conn. $T = 250\,\text{ms}$	x	x	2227	3.01
Monitoring $T = 3000\,\text{ms}$	57	0.07	57	0.08
state LED	33090	41.44	26400	35.66
battery LED	26400	33.06	26400	35.66
Total	**79852**	100.00	**74025**	100.00
Total $t_{10\%}$ [d]	**5.01**		**5.40**	

Table 9.6: Battery consumption - system states

Deep Sleep Consumption

The test of static consumption (9.1.1) showed that it is possible to reach a very low average current when the MCU is sleeping (around 8 µA). It is an improvement in comparison with the older SCILIF light control unit, for which an average current of 23 µA was measured when sleeping. The old unit is based on STMF051K4 with datasheet current consumption of 3.7 µA [33, p. 52] in the most power-saving mode.

RDM6300 vs LC RFID Reader

It was found that the external RFID reader RDM6300 can not meet the requirement on battery life. It would drain 10 % of the 1600 mAh battery in 10.4 hours. On the contrary, the LC RFID reader implemented directly on board consumes the same energy in approximately 21 days. Furthermore, this duration can be even prolonged by adjusting the parameters - the period of checking or the number of samples.

TX Power Levels & RX/TX Amplitudes

Considering the ratio of BLE activity in the overall consumption (table 9.6), the test of TX power levels clearly showed that only a very insignificant amount of the overall energy, roughly tenths of a percent, can be saved by decreasing the level from the default 0 dBm to -4 dBm or lower. On the other hand, it significantly reduces the possible communication range (see table 9.8).

Surprisingly, the measured RX/TX current amplitudes of either advertising or data packets at 0 dBm are much higher than those proclaimed in the nRF52 datasheet (RX: 4.6 mA vs measured 12.3 mA; TX: 4.8 mA vs measured 8 mA).

RFID Check and Detection Optimization

Checking the presence of RFID tag proved to be one of the biggest consumers of battery energy (25 %). To improve it, the period of checking can be increased or fewer samples could be taken. Nonetheless, increasing the period reduces the responsivity and user experience.
Additionally, the detection of RFID tag word is by far the largest energy consumer among all measured events. The figure 9.4 illustrates that the processing takes a relatively small part of the overall energy, so there is not much space for improvement. Nevertheless, when dealing with false positives (check positive, detection running, but tag not present), it would be beneficial to sample smaller partitions and process them online. This way, it would be possible to quit the sampling and processing earlier and save energy.

Advertising and Connection Interval

Furthermore, increasing the advertising or connection interval (period) does not bring much energy savings due to the small ratio of BLE activity in the overall consumption (table 9.6). On the other hand, it notably reduces the user experience while using the mobile application, mostly because of increased loading times.

Usage of LEDs

Lastly, but most importantly, the table 9.6 shows that the indication LEDs consume more than 70 % of overall energy. From the user's perspective, it is important to know the unit's state by some form of indication.

Still, in order to optimize the overall battery life of the light control unit, this is the starting point for further analysis of a more energy-saving user interface. Using the indication LEDs more wisely could potentially result in much better consumption than approximately five days (consumption of 10 % of battery).

9.2 Communication Range

Multiple measurements were performed to test the communication range of the light control unit - nRF52 radio. The goal of the test was not to provide precise values but to give an estimate of what the communication range might look like.

The test was performed outdoor (direct line of sight) with two mobile devices, Android-based Huawei Honor 8 and iOS-based Apple iPhone 8, connected to a light control unit. A person holding a smartphone always tested the communication and moved in 2-5 meter intervals until a connection dropout occurred or the latency between R/W operations was too high.

The communication range was measured for three different light control units using the default TX output power level of 0 dBm, once without and once with 2.4 GHz, 2.4 dBi antenna. For one of the light control units, different TX power levels were configured and the range measured.

Light Control Unit:	**LCU 1**	**LCU 2**	**LCU 3**
No antenna, range [m]:	5	2.5	4
Antenna, range [m]:	53	33	40

LCU1 - TX Power [dBm]:	+8	+4	0	-4	-8	-12	-16
LCU1 - Range [m]:	70	62	53	25	22	7.5	5

Table 9.8: Range test results

Conclusions

The table 9.7 shows the measured values. The problem with the test is that while testing the communication range of nRF52 on the light control unit, no reference - RX/TX parameters of the counterpart was established, so this test is not reproducible. The react library react-native-ble-plx used in the mobile application (see section 7.2.1) does not provide an API to set or get the TX power level or sensitivity of the receiver. However, there was no significant difference in communication range between the mobile devices. iPhone usually lost signal a few seconds before Android.

A surprising fact is that the measured communication range differs hugely among light control units. It might be caused by the differences in the quality of antennas or their polarization.

9.3 Regulation of LED Current

To control the current flowing through the light emitting diode that illuminates the light guide, the hardware regulation based on DIO5611 chip and SEPIC converter was used. The required value of current is set through PWM duty cycle (see section 4.1.4).

The LED current was measured for all possible PWM duty cycles with varying input voltage to evaluate the linearity of the conversion function. The measured results were linearly fitted and compared with the ideal characteristic (see section 4.1.4).

Input Voltage [V]	**RMSE [mA]**	**Slope Error [%]**
2.8	0.316	0.383
3.0	0.283	0.136
3.2	0.298	0.047
3.4	0.299	0.088
3.6	0.320	0.070
3.8	0.303	0.131
4.0	0.313	0.117
4.2	0.326	0.364

Table 9.9: LED regulation results

Conclusions

Table 9.9 shows the root mean square error and relative error of characteristic slope for eight different input voltages. According to the results, the regulation works as expected with very low slope error and root mean square error (less than 1 mA) regardless of input voltage. Considering the fact that the human eye is not sensitive enough to spot such subtle differences, this regulation precision is satisfying.

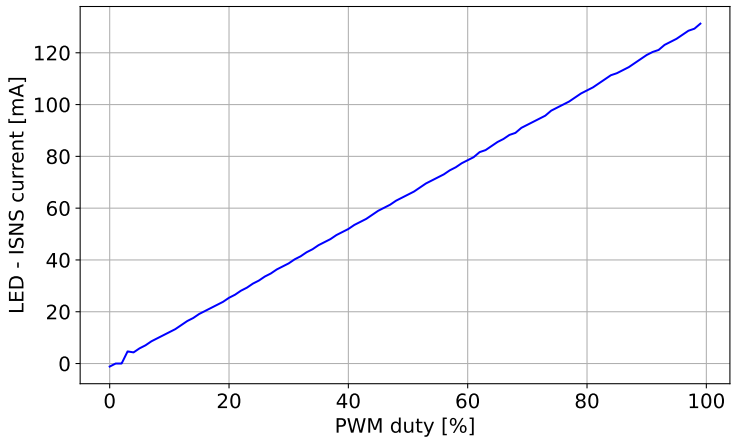


Figure 9.5: Measured characteristic for input voltage 3.6 V

9.4 RFID Detection Range

Considering the typical use case when the RFID tag is encased in clothing, a communication range of a few centimeters would fulfil the requirements.

The detection range was tested with two different RFID tag types - a very cheap EM4100 tag directly from China similar to the one in figure 4.11 and a bigger EM4100 tag with better sensitivity used in an access control system.

Conclusions

In both cases, it was almost impossible to measure precisely the detection range. It depends hugely on the mutual orientation of the reader's coil and the tag and the surrounding environment. An average detection range varied from 2 to 4 centimeters.

9.5 A+A Düsseldorf

The ongoing development (light control unit and mobile application) was successfully presented alongside other SCILIF products at the A+A exhibition in Düsseldorf, Germany. A+A is one of the biggest international trade fairs covering the entire spectrum of exhibits for safety and security.

My own pictures from the exhibition are shown in appendix A.1.

Chapter 10

Conclusion

This thesis describes an analysis, design and development of the new complete platform around remote control of active lighting system from SCILIF company. The source code was maintained through git versioning system (available here).

Light control unit, a battery device based on nRF52 BLE SoC, was designed to regulate the luminance of a light guide in predefined light modes, monitor the device condition and control the Bluetooth connection (advertising, connection, bonding etc.). Based on the analysis of possible remote control methods, trade-offs between practicality, technical complexity and long battery life requirement were made to finally select local control by board button, contactless activation by RFID tag, remote control by BLE via a smartphone application or a floating button, or gesture control. With the exception of gesture control, for which the development only started, all of the methods above were fully incorporated into the device firmware.
Moreover, a lot of effort was given to the light control unit's user interface design, which is based on only one push-button and three indication LEDs, to allow all user interactions.

At first, the **detection of RFID** tags was tested with an external RFID reader whose biggest drawback was large standby current. Consequently, a second custom RFID reader, based on PWM excitation of LC oscillator with synchronous sampling of RFID signal carrier, was integrated directly into the light control unit. As a result, not only the detection range (up to 4 centimeters) and the latency (less than 200 ms), but also the battery consumption was improved. Draining 10 % of the battery in no less than 21 days instead of a few hours was the biggest enhancement of the custom solution over the external reader.

For the **mobile application**, the React Native framework facilitating the application building on both mobile platforms, Android and iOS, was preferred. Multiple users installed and tested the application locally. The iOS users via a special third-party platform TestFlight which is available only through a paid Apple Developer Program.
The application allows the user to scan, communicate with one or multiple light control units and read or write data through three custom BLE services - LED

Control, Monitor and RFID service. Changing the light mode, configuring the RFID detection such as paired tag ID, or monitoring the device condition (battery level, temperature, charging status, light guide properties etc.) are the possible user interactions. The application also enables upgrading the firmware over-the-air (OTA upgrade) for both the light control unit and floating button through a standalone firmware program - bootloader.

The **floating button** is an external BLE based button with good haptic properties for fast and comfortable light mode change. However, so far, it was implemented only as a prototype on the micro:bit v.2 development board. From the BLE perspective, its implementation was more complex because it can communicate as BLE central and peripheral even simultaneously.

Finally, multiple **tests of results** with the light control unit were performed. The test of communication range showed that a light control unit with an external antenna is able to communicate up to 50 meters with a smartphone in case of a direct line of sight. Validating the linearity of LED current regulation, and thus light guide luminance verified that its error is far better than the sensitivity of the human eye.
The comprehensive analysis of battery consumption provided some interesting results. Because of code optimization and selection of the low power components, the unit consumes around 8 µA in the most power-saving mode. Surprisingly, the effect of BLE transmissions on the overall consumption is very low (1 - 3 %), in comparison with the indication by LED diodes which proved to be the biggest energy consumer (around 75 %).
The ongoing development was also successfully presented at the A+A exhibition around safety and security in Düsseldorf, Germany.

Future Work

There are always multiple things to improve in future. Apart from smaller tasks, these jobs would have the highest priority.

Incorporation of an RTOS
Handling various scheduled tasks and asynchronous events became sometimes unreliable without a proper real-time operating system. For example, NuttX could be used in our case.

Finishing Gesture Control
There is a huge space for future development around gesture control on embedded devices. Its detailed analysis and implementation could be a topic for another thesis.

Migration to Native Mobile Platforms
Using React Native brought many benefits, especially fast development, on the other hand, a lot of trade-offs too. Not having the ability to control the BLE stack directly was very limiting. One of the future goals is to implement a smartwatch application on iWatch. Considering that React has no direct support, and most of the application logic will be shared with the smartphone application, it might be worthwhile to migrate to the native programming tools (Swift).

Appendix A

Additional Figures & Schemes

Figure A.1: SCILIF stall at A+A Düsseldorf

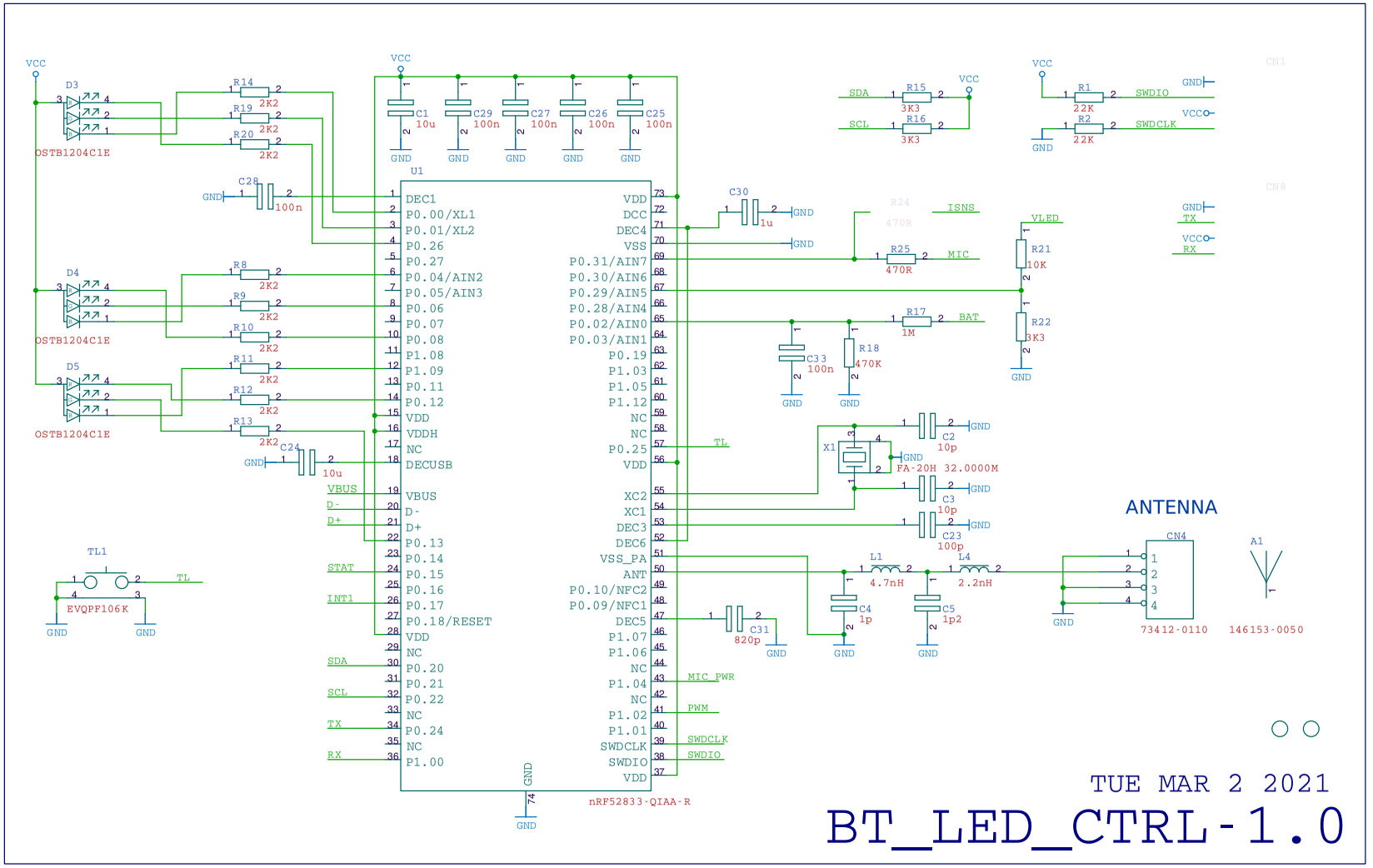


Figure A.2: The HW design of light control unit board (created by Petr Porazil)

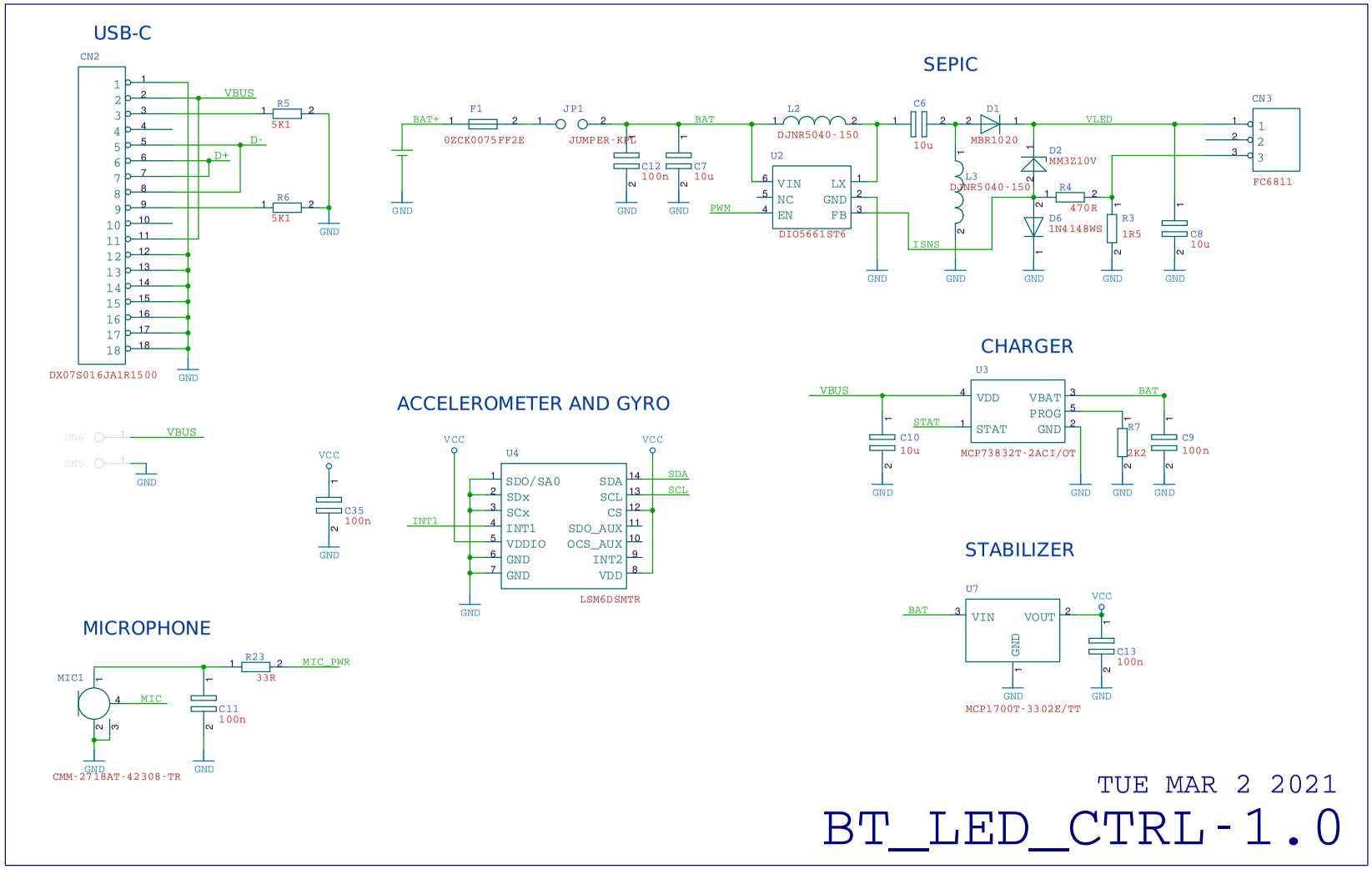


Figure A.3: The HW design of light control unit board (created by Petr Porazil)

Appendix B

List of Acronyms and Symbols

Shortcut	Meaning
Acc	Accelerometer
AD	Advertising Data
ADC	Analog-to-Digitial Convertor
API	Application Programming Interface
ASK	Amplitude Shift Keying
ATT	ATTribute Protocol
BLE	Bluetooth Low Energy
Bluetooth SIG	Bluetooth Special Interest Group
BSP	Board Support Package
CCCD	Client Characteristic Configuration Descriptor
CRC	Cyclic Redundancy Check
CMOS	Complementary Metal-Oxide Semiconductor
DFU	Device Firmware Upgrade
DMA	Direct Memory Access
FB	Floating Button
FPU	Floating Point Unit
FSK	Frequency Shift Keying
GAP	Generic Access Profile
GATT	Generic ATTribute profile
GFSK	Gaussian Frequency Shift Keying
GPIO	General Purpose Input Output
HAL	Hardware Abstract Layer
HF	High Frequency
HFCLK	High Frequency Clock
IRQ	Interrupt Request
ISM	Industrial, Scientific and Medical
ISR	Interrupt Service Routine
I^2C	Inter-Integrated Circuit
LCU	Light Control Unit
LED	Light Emitting Diode
LESC	LE Secure Connection

Shortcut	Meaning
LF	Low Frequency
LSB	Least Significant Bit
LTK	Long Term Key
MAC	Media Access Control
MBR	Master Boot Record
MCU	MicroController Unit
MEMS	Micro-Electromechanical System
MTIM	Man In the Middle
MTU	Maximum Transmission Unit
NRZ	Non Return to Zero
ODR	Output Data Rate
OTA	Over The Air
RAM	Random Access Memory
PDU	Protocol Data Unit
PPI	Programmable Peripheral Interconnect
PSK	Phase Shift Keying
PWM	Pulse Width Modulation
RFID	Radio Frequency IDentification
SDK	Software Development Kit
SEPIC	Single-Ended Primary Inductor Convertor
SM	Security Manager
SoC	System on Chip
SPI	Serial Peripheral Interface
STK	Short Term Key
SWD	Serial Wire Debug
RAM	Randim Access Memory
RISC	Reduced Instruction Set Computer
RSSI	Received Signal Strength Indicator
RTC	Real-Time Clock
RTOS	Real-Time Operating System
TTL	Transistor–Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
UHF	Ultra-High Frequency
USB	Universal Serial Bus
UUID	Universally Unique IDentifier

Appendix C

Bibliography

[3] *DIO5661 Datasheet*. English. [Online; accessed 10-04-2022 from https://datasheetspdf.com/pdf-file/1351599/DIOOMICROCIRCUITS/DIO5661/1]. Dioo Microcircuits co., Ltd. Apr. 2016.

[4] *EM4100 Datasheet*. English. [Online; accessed 11-04-2022 from https://www.alldatasheet.com/datasheet-pdf/pdf/154654/EMMICRO/EM4100.html]. EM Microelectronic. Feb. 2004.

[5] Core Specification Working Group. *Bluetooth Core Specification*. English. Version Revision 5.2. Bluetooth SIG. Dec. 2019.

[6] Hobeom Han and Sang Won Yoon. "Gyroscope-Based Continuous Human Hand Gesture Recognition for Multi-Modal Wearable Input Device for Human Machine Interaction". In: *Sensors* 19.11 (2019). DOI: 10.3390/s19112562.

[7] V. Daniel Hunt and Albert Puglia. *RFID A Guide to Radio Frequency Identification*. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2007.

[8] *LSM303AGR Ultra-compact high-performance eCompass module*. English. Version Rev. 10. [Online; accessed 11-05-2022 from https://www.st.com/en/mems-and-sensors/lsm303agr.html]. STMicroelectronics. Nov. 2018.

[9] *LSM6DSM iNEMO inertial module Datasheet*. English. Version Rev. 7. [Online; accessed 11-05-2022 from https://www.st.com/en/mems-and-sensors/lsm6dsm.html]. STMicroelectronics. Oct. 2017.

[10] *MCP1700 Datasheet*. English. [Online; accessed 11-05-2022 from https://www.microchip.com/en-us/product/MCP1700]. Microchip Technology Inc. Jan. 2005.

[11] *MCP73831/2 Datasheet*. English. [Online; accessed 10-01-2022 from https://ww1.microchip.com/downloads/en/DeviceDoc/MCP73831-Family-Data-Sheet-DS20001984H.pdf]. Microchip Technology Inc. Feb. 2020.

[12] *Measuring Lithium Battery Voltage*. https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/measuring-lithium-battery-voltage-with-nrf52. [Online; accessed 27-02-2022]. May 2016.

[14] *nRF SDK Documentation - Bootloader*. English. Version Revision v17.0.2. [Online; accessed 23-02-2022 from https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/lib_bootloader.html]. Nordic Semiconductor. Sept. 2020.

[15] *nRF SDK Documentation - Buttonless Secure DFU Service*. English. Version Revision v17.0.2. [Online; accessed 09-02-2022 from https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/service_dfu.html]. Nordic Semiconductor. Sept. 2020.

[16] *nRF SDK Documentation - Device Firmware Update process*. English. Version Revision v17.0.2. [Online; accessed 09-02-2022 from https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/lib_bootloader_dfu_process.html]. Nordic Semiconductor. Sept. 2020.

[17] *nRF SDK Documentation - DFU BLE Service*. English. Version Revision v17.0.2. [Online; accessed 09-02-2022 from https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.0/lib_dfu_transport_ble.html]. Nordic Semiconductor. Sept. 2020.

[18] *nRF SDK Documentation - Secure Bootloader*. English. Version Revision v17.0.2. [Online; accessed 09-02-2022 from https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/ble_sdk_app_secure_bootloader.html]. Nordic Semiconductor. Sept. 2020.

[19] *nRF SDK Documentation - SoftDevices*. English. Version Revision v17.0.2. [Online; accessed 23-02-2022 from https://infocenter.nordicsemi.com/topic/struct_nrf52/struct/nrf52_softdevices.html]. Nordic Semiconductor. Feb. 2022.

[24] *nRF52833 Product Specification*. English. Version 1.5. [Online; accessed 10-01-2022 from https://infocenter.nordicsemi.com/topic/ps_nrf52833/keyfeatures_html5.html?cp=4_1_0]. Nordic Semiconductor. Nov. 2021.

[26] *Personal Safety: Illumination In The 21st Century*. https://www.ehstoday.com/construction/article/21916116/personal-safety-illumination-in-the-21st-century. [Online; accessed 27-02-2022]. Feb. 2014.

[27] *Používané RFID frekvence a jejich vliv na čtení a zápis tagu*. https://automatizace.hw.cz/komponenty-prumyslove-sbernice-a-komunikace/vice-i-mene-bezne-rfid-frekvence-a-jejich-vliv-na-vlastnosti-tagu.html. [Online; accessed 01-05-2022]. Sept. 2015.

[28] *React Native*. https://medium.com/@thinkwik/react-native-what-is-it-and-why-is-it-used-b132c3581df. [Online; accessed 20-04-2022]. Jan. 2018.

[33] *STM32F051K4 Datasheet*. English. Version Rev. 7. [Online; accessed 11-05-2022 from https://www.st.com/en/microcontrollers-microprocessors/stm32f051k4.html]. STMicroelectronics. Jan. 2017.

[34] *Switch Bounce and How to Deal with It*. https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/. [Online; accessed 25-02-2022]. Sept. 2015.

[36] Kewin Townsend, Carles Cufí, and Akiba & Robery Davidson. *Getting Started with Bluetooth Low Energy*. Sebastopol, California, USA: OŔeilly Media, Inc, May 2014.

[37] *Types of Mobile Application Development: Native, Hybrid, Platform-Based Approaches, and More*. https://medium.com/yellow-universe/types-of-mobile-application-development-native-hybrid-platform-based-approaches-and-more-29a2322fcb15. [Online; accessed 20-04-2022]. July 2020.

[38] A. Weiskopf, F. Weichert, and E. Dommes. "A low cost embedded solution for detection and analysis of three-dimensional gestures through wireless communication". In: *IFAC Proceedings Volumes* 45.4 (2012). 1st IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control, pp. 272–277. DOI: https://doi.org/10.3182/20120403-3-DE-3010.00034.

[39] *What is Component Diagram?* https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/. [Online; accessed 20-04-2022].

Appendix D

Sources - Images

[1] *125khz RFID Reader Module RDM6300*. https://cityelectronics.pk/shop/125khz-rfid-reader-module-rdm6300-arduino-compatible/. [Online; accessed 22-02-2022].

[2] *BLE Advertising Primer*. https://www.argenox.com/library/bluetooth-low-energy/ble-advertising-primer/. [Online; accessed 24-04-2022]. Feb. 2019.

[13] *Microbit Board*. https://microbit.org/. [Online; accessed 22-02-2022].

[16] *nRF SDK Documentation - Device Firmware Update process*. English. Version Revision v17.0.2. [Online; accessed 09-02-2022 from https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/lib_bootloader_dfu_process.html]. Nordic Semiconductor. Sept. 2020.

[22] *nRF52 Memory Layout*. https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/lib_bootloader.html?cp=8_5_0_3_5_0. [Online; accessed 24-02-2022]. Sept. 2021.

[23] *nRF52833 Block Diagram*. https://infocenter.nordicsemi.com/topic/ps_nrf52833/blockdiagram.html?cp=4_1_0_21. [Online; accessed 22-02-2022]. Apr. 2021.

[25] *nRF52833 Radio Block Diagram*. https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.nrf52832.ps.v1.1/radio.html?cp=4_2_0_22#concept_lhd_ygj_4r. [Online; accessed 22-02-2022]. Oct. 2021.

[30] *RFID Technology: System, Principle, Classification and Application*. https://www.apogeeweb.net/article/136.html. [Online; accessed 24-04-2022]. Nov. 2018.

[32] *Schematic of SEPIC*. https://en.wikipedia.org/wiki/Single-ended_primary-inductor_converter. [Online; accessed 02-05-2022]. Aug. 2006.

[35] *TK4100 125KHz RFID key fob*. https://www.neven.cz/kategorie/elektronicke-soucastky/nfc-a-rfid/rfid/tk4100-125khz-rfid-neprogramovatelny-cip-klicenka-modry/. [Online; accessed 22-02-2022].

Appendix E

Sources - Software

[20] *nRF Util*. [Online; accessed 09-05-2022 from `https://github.com/NordicSemiconductor/pc-nrfutil`].

[21] *nRF5 SDK*. [Online; accessed 09-05-2022 from `https://www.nordicsemi.com/Products/Development-software/nRF5-SDK`].

[29] *React Native BLE Plx*. [Online; accessed 09-05-2022 from `https://github.com/dotintent/react-native-ble-plx`].

[31] *RigolWFM Python Library*. [Online; accessed 09-05-2022 from `https://pypi.org/project/RigolWFM/`].